

ENOVIA Live Collaboration™

V6R2012



Application Development Guide

Copyright and Trademark Information

© Dassault Systèmes, 2000 - 2011

All rights reserved.

PROPRIETARY RIGHTS NOTICE: This documentation is proprietary property of Dassault Systèmes and its subsidiaries. This documentation shall be treated as confidential information and may only be used by employees or contractors with the Customer in accordance with the applicable Software License Agreement.

Adapt@®, Compliance Connect®, DesignSync®, ENOVIA®, MatrixOne®, ProjectSync®, Synchronicity®, and Team Central® are registered trademarks of Dassault Systèmes.

ENOVIA Live Collaboration, ENOVIA Live Collaboration Business Process Services, ENOVIA Live Collaboration Server, ENOVIA Studio Modeling Platform, ENOVIA Studio Federation Toolkit, ENOVIA Studio Customization Toolkit, ENOVIA 3D Live, ENOVIA Engineering Central, ENOVIA Library Central, ENOVIA Material Compliance Central, ENOVIA Program Central, ENOVIA Sourcing Central, ENOVIA Specification Central, ENOVIA Supplier Central, ENOVIA Design Central, ENOVIA Collaborative Interference Management, ENOVIA Semiconductor Accelerator for Team Compliance, ENOVIA Aerospace and Defense Accelerator for Program Management, ENOVIA Apparel Accelerator for Design and Development, ENOVIA X-BOM Cost Analytics, ENOVIA X-BOM Manufacturing, ENOVIA Variant Configuration Central, ENOVIA Synchronicity DesignSync Data Manager, IconMail, ImageIcon and Star Browser are trademarks of Dassault Systèmes.

Oracle® is a registered trademark of Oracle Corporation, Redwood City, California. DB2, AIX, and WebSphere are registered trademarks of IBM Corporation. WebLogic is a registered trademark of BEA Systems, Inc. Solaris, UltraSPARC, Java, JavaServer Pages, JDBC, and J2EE are registered trademarks of Sun Microsystems, Inc. Windows XP and Internet Explorer are registered trademarks of Microsoft Corp. HP and HP-UX are registered trademarks of HP. All other product names and services identified throughout this book are recognized as trademarks, registered trademarks, or service marks of their respective companies.

The documentation that accompanies ENOVIA products describes the applications as delivered by Dassault Systèmes. This documentation includes readme files, online help, user guides, and administrator guides. If changes are made to an application or to the underlying framework, Dassault Systèmes cannot ensure the accuracy of this documentation. These changes include but are not limited to: changing onscreen text, adding or removing fields on a page, making changes to the administrative objects in the schema, adding new JSPs or changing existing JSPs, changing trigger programs, changing the installation or logon process, or changing the values in any property file. For instructions on customizing the provided documentation, see the *ENOVIA Live Collaboration Administrator's Guide*.

Dassault Systems Enovia Corp.
900 Chelmsford Street
Lowell, MA 01851

Telephone 978.442.2500
Email: enovia.info@3ds.com

<http://www.3ds.com>

Additional Components

This product also includes additional components copyrighted by other third parties. The sections that follow provide license and copyright notices of these software components.

Apache

Apache License

Version 2.0, January 2004

<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor, for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License;

(b) You must cause any modified files to carry prominent notices stating that You changed the files;

(c) You must retain in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranty or condition of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may only act on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Apache Ant

=====

NOTICE file corresponding to the section 4 d of the Apache License, Version 2.0, in this case for the Apache Ant distribution.

=====

This product includes software developed by The Apache Software Foundation (<http://www.apache.org/>).

This product includes also software developed by :

- the W3C consortium (<http://www.w3c.org/>) ,
- the SAX project (<http://www.saxproject.org>)

Please read the different LICENSE files present in the root directory of this distribution. [BELOW]

This license came from:

<http://www.w3.org/Consortium/Legal/copyright-software-19980720>

W3C® SOFTWARE NOTICE AND LICENSE

Copyright © 1994-2001 World Wide Web Consortium, World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

<http://www.w3.org/Consortium/Legal/>

This W3C work (including software, documents, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions: Permission to use, copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications, that you make:

The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.

Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, a short notice of the following form (hypertext is preferred, text is permitted) should be used within the body of any redistributed or derivative code:

'Copyright © 1999-2004 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. <http://www.w3.org/Consortium/Legal/>'

Notice of any changes or modifications to the W3C files, including the date changes were made. (We recommend you provide URLs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS." AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

This license came from: <http://www.megginson.com/SAX/copying.html>. However please note future versions of SAX may be covered under <http://saxproject.org/?selected=pdf>

This page is now out of date -- see the new SAX site at <http://www.saxproject.org/> for more up-to-date releases and other information. Please change your bookmarks.

SAX2 is Free!

I hereby abandon any property rights to SAX 2.0 (the Simple API for XML), and release all of the SAX 2.0 source code, compiled code, and documentation contained in this distribution into the Public Domain. SAX comes with NO WARRANTY or guarantee of fitness for any purpose.

David Megginson, david@megginson.com

Apache Axis

=====

NOTICE file corresponding to section 4(d) of the Apache License, Version 2.0, in this case for the Apache Axis distribution.

=====

This product includes software developed by The Apache Software Foundation (<http://www.apache.org/>).

Apache Tomcat

[under Apache License, Version 2.0 above]

Apache Servlet-API

[under Apache License, Version 2.0 above]

FTP

Copyright (c) 1983, 1985, 1989, 1993, 1994

The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1997-1999 The Stanford SRP Authentication Project

All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL STANFORD BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER

Copyright 1990 by the Massachusetts Institute of Technology.

All Rights Reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Getline

Copyright (C) 1991, 1992, 1993 by Chris Thewalt (thewalt@ce.berkeley.edu)

Permission to use, copy, modify, and distribute this software for any purpose and without fee is hereby granted, provided that the above copyright notices appear in all copies and that both the copyright notice and this permission notice appear in supporting documentation. This software is provided "as is" without express or implied warranty.

GifEncoder

GifEncoder - write out an image as a GIF

Transparency handling and variable bit size courtesy of Jack Palevich.

Copyright (C)1996,1998 by Jef Poskanzer <jef@acme.com>. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ImageEncoder

ImageEncoder - abstract class for writing out an image

Copyright (C) 1996 by Jef Poskanzer <jef@acme.com>. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JavaMail

Sun Microsystems, Inc.

Binary Code License Agreement

READ THE TERMS OF THIS AGREEMENT AND ANY PROVIDED SUPPLEMENTAL LICENSE TERMS (COLLECTIVELY "AGREEMENT") CAREFULLY BEFORE OPENING THE SOFTWARE MEDIA PACKAGE. BY OPENING THE SOFTWARE MEDIA PACKAGE, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCESSING THE SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY RETURN THE UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A REFUND OR, IF THE SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT THE "DECLINE" BUTTON AT THE END OF THIS AGREEMENT.

1. LICENSE TO USE. Sun grants you a non-exclusive and non-transferable license for the internal use only of the accompanying software and documentation and any error corrections provided by Sun (collectively "Software"), by the number of users and the class of computer hardware for which the corresponding fee has been paid.

2. RESTRICTIONS. Software is confidential and copyrighted. Title to Software and all associated intellectual property rights is retained by Sun and/or its licensors. Except as specifically authorized in any Supplemental License Terms, you may not make copies of Software, other than a single copy of Software for archival purposes. Unless enforcement is prohibited by applicable law, you may not modify, decompile, or reverse engineer Software. You acknowledge that Software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Sun disclaims any express or implied warranty of fitness for such uses. No right, title or interest in or to any trademark, service mark, logo or trade name of Sun or its licensors is granted under this Agreement.

3. LIMITED WARRANTY. Sun warrants to you that for a period of ninety (90) days from the date of purchase, as evidenced by a copy of the receipt, the media on which Software is furnished (if any) will be free of defects in materials and workmanship under normal use. Except for the foregoing, Software is provided "AS IS". Your exclusive remedy and Sun's entire liability under this limited warranty will be at Sun's option to replace Software media or refund the fee paid for Software.

4. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

5. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun's liability to you, whether in contract, tort (including negligence), or otherwise, exceed the amount paid by you for Software under this Agreement. The foregoing limitations will apply even if the above stated warranty fails of its essential purpose.

6. Termination. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of Software. This Agreement will terminate immediately without notice from Sun if you fail to comply with any provision of this Agreement. Upon Termination, you must destroy all copies of Software.

7. Export Regulations. All Software and technical data delivered under this Agreement are subject to US export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain such licenses to export, re-export, or import as may be required after delivery to you.

8. U.S. Government Restricted Rights. If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

9. Governing Law. Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

10. Severability. If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

11. Integration. This Agreement is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgement, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

JAVAMAIL™, VERSION 1.3.1

SUPPLEMENTAL LICENSE TERMS

These supplemental license terms ("Supplemental Terms") add to or modify the terms of the Binary Code License Agreement (collectively, the "Agreement"). Capitalized terms not defined in these Supplemental Terms shall have the same meanings ascribed to them in the Agreement. These Supplemental Terms shall supersede any inconsistent or conflicting terms in the Agreement, or in any license contained within the Software.

1. Software Internal Use and Development License Grant. Subject to the terms and conditions of this Agreement, including, but not limited to Section 3 (Java(TM) Technology Restrictions) of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license to reproduce internally and use internally the binary form of the Software, complete and unmodified, for the sole purpose of designing, developing and testing your Java applets and applications ("Programs").

2. License to Distribute Software. Subject to the terms and conditions of this Agreement, including, but not limited to Section 3 (Java(TM) Technology Restrictions) of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license to reproduce and distribute the Software to third party software developers for the purpose of developing additional software which invokes such additional API, you must promptly publish in binary code form, provided that (i) you distribute the Software complete and unmodified and only bundled as part of, and for the sole purpose of running, your Java applets or applications ("Programs"), (ii) the Programs add significant and primary functionality to the Software, (iii) you do not distribute additional software intended to replace any component(s) of the Software, (iv) you do not remove or alter any proprietary legends or notices contained in the Software, (v) you only distribute the Software subject to a license agreement that protects Sun's interests consistent with the terms contained in the Agreement, and (vi) you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

3. Java Technology Restrictions.* You may not modify the Java Platform Interface ("JPI", identified as classes contained within the "java" package or any subpackages of the "java" package), by creating additional classes within the JPI or otherwise causing the addition to or modification of the classes in the JPI. In the event that you create an additional class and associated API(s) which (i) extends the functionality of the Java platform, and (ii) is exposed to third party software developers for the purpose of developing additional software which invokes such additional API, you must promptly publish broadly an accurate specification for such API for free use by all developers. You may not create, or authorize your licensees to create additional classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Sun in any naming convention designation.

4. Trademarks and Logos. You acknowledge and agree as between you and Sun that Sun owns the SUN, SOLARIS, JAVA, JINI, FORTE, STAROFFICE, STARPORTAL and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, STAROFFICE, STARPORTAL, and iPLANET-related trademarks, service marks, logos and other brand designations ("Sun Marks"), and you agree to comply with the Sun Trademark and Logo Usage Requirements currently located at <http://www.sun.com/policies/trademarks>. Any use you make of the Sun Marks inures to Sun's benefit.

5. Source Code. Software may contain source code that is provided solely for reference purposes pursuant to the terms of this Agreement. Source code may not be redistributed unless expressly provided for in this Agreement.

6. Termination for Infringement. Either party may terminate this Agreement immediately should any Software become, or in either party's opinion be likely to become, the subject of a claim of infringement of any intellectual property right.

For inquiries please contact: Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A.

(LF#132726/Form ID#011801)

Java POI

[under Apache License, Version 2.0 above]

JDK

Sun Microsystems, Inc. Binary Code License Agreement

for the JAVA 2 PLATFORM STANDARD EDITION DEVELOPMENT KIT 5.0

SUN MICROSYSTEMS, INC. ("SUN") IS WILLING TO LICENSE THE SOFTWARE IDENTIFIED BELOW TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS BINARY CODE LICENSE AGREEMENT AND SUPPLEMENTAL LICENSE TERMS (COLLECTIVELY "AGREEMENT"). PLEASE READ THE AGREEMENT CAREFULLY. BY DOWNLOADING OR INSTALLING THIS SOFTWARE, YOU ACCEPT THE TERMS OF THE AGREEMENT. INDICATE ACCEPTANCE BY SELECTING THE "ACCEPT" BUTTON AT THE BOTTOM OF THE AGREEMENT. IF YOU ARE NOT WILLING TO BE BOUND BY ALL THE TERMS, SELECT THE "DECLINE" BUTTON AT THE BOTTOM OF THE AGREEMENT AND THE DOWNLOAD OR INSTALL PROCESS WILL NOT CONTINUE.

1. DEFINITIONS. "Software" means the identified above in binary form, any other machine readable materials (including, but not limited to, libraries, source files, header files, and data files), any updates or error corrections provided by Sun, and any user manuals, programming guides and other documentation provided to you by Sun under this Agreement. "Programs" mean Java applets and applications intended to run on the Java 2 Platform Standard Edition (J2SE) platform) platform on Java-enabled general purpose desktop computers and servers.

2. LICENSE TO USE. Subject to the terms and conditions of this Agreement, including, but not limited to the Java Technology Restrictions of the Supplemental License Terms, Sun grants you a non-exclusive, non-transferable, limited license without license fees to reproduce and use internally Software complete and unmodified for the sole purpose of running Programs. Additional licenses for developers and/or publishers are granted in the Supplemental License Terms.

3. RESTRICTIONS. Software is confidential and copyrighted. Title to Software and all associated intellectual property rights is retained by Sun and/or its licensors. Unless enforcement is prohibited by applicable law, you may not modify, decompile, or reverse engineer Software. You acknowledge that Licensed Software is not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Sun Microsystems, Inc. discloses any express or implied warranty of fitness for such uses. No right, title or interest in or to any trademark, service mark, logo or trade name of Sun or its licensors is granted under this Agreement. Additional restrictions for developers and/or publishers licenses are set forth in the Supplemental License Terms.

4. LIMITED WARRANTY. Sun warrants to you that for a period of ninety (90) days from the date of purchase, as evidenced by a copy of the receipt, the media on which Software is furnished (if any) will be free of defects in materials and workmanship under normal use. Except for the foregoing, Software is provided "AS IS". Your exclusive remedy and Sun's entire liability under the limited warranty will be at Sun's option to replace Software media or refund the fee paid for Software. Any implied warranties on the Software are limited to 90 days. Some states do not allow limitations on duration of an implied warranty, so the above may not apply to you. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

5. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

6. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun's liability to you, whether in contract, tort (including negligence), or otherwise, exceed the amount paid by you for Software under this Agreement. The foregoing limitations will apply even if the above stated warranty fails of its essential purpose. Some states do not allow the exclusion of incidental or consequential damages, so some of the terms above may not be applicable to you.

7. TERMINATION. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of Software. This Agreement will terminate immediately without notice from Sun if you fail to comply with any provision of this Agreement. Either party may terminate this Agreement immediately should any Software become, or in either party's opinion be likely to become, the subject of a claim of infringement of any intellectual property right. Upon Termination, you must destroy all copies of Software.

8. EXPORT REGULATIONS. All Software and technical data delivered under this Agreement are subject to US export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain such licenses to export, re-export, or import as may be required after delivery to you.

9. TRADEMARKS AND LOGOS. You acknowledge and agree as between you and Sun that Sun owns the SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET-related trademarks, service marks, logos and other brand designations ("Sun Marks"), and you agree to comply with the Sun Trademark and Logo Usage Requirements currently located at <http://www.sun.com/policies/trademarks>. Any use you make of the Sun Marks inures to Sun's benefit.

10. U.S. GOVERNMENT RESTRICTED RIGHTS. If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

11. GOVERNING LAW. Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

12. SEVERABILITY. If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

13. INTEGRATION. This Agreement is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgement, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

SUPPLEMENTAL LICENSE TERMS

These Supplemental License Terms add to or modify the terms of the Binary Code License Agreement. Capitalized terms not defined in these Supplemental Terms shall have the same meanings ascribed to them in the Binary Code License Agreement . These Supplemental Terms shall supersede any inconsistent or conflicting terms in the Binary Code License Agreement, or in any license contained within the Software.

A. Software Internal Use and Development License Grant. Subject to the terms and conditions of this Agreement and restrictions and exceptions set forth in the Software "README" file, including, but not limited to the Java Technology Restrictions of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license without fees to reproduce internally and use internally the Software complete and unmodified for the purpose of designing, developing and testing your Programs.

B. License to Distribute Software. Subject to the terms and conditions of this Agreement and restrictions and exceptions set forth in the Software README file, including, but not limited to the Java Technology Restrictions of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license without fees to reproduce and distribute the Software to third party software developers for the purpose of developing additional software which invokes such additional API, you must promptly publish in binary code form, provided that (i) you distribute the Software complete and unmodified and only bundled as part of, and for the sole purpose of running, your Programs, (ii) the Programs add significant and primary functionality to the Software, (iii) you do not distribute additional software intended to replace any component(s) of the Software, (iv) you do not remove or alter any proprietary legends or notices contained in the Software, (v) you only distribute the Software subject to a license agreement that protects Sun's interests consistent with the terms contained in the Agreement, and (vi) you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

C. License to Distribute Redistributables. Subject to the terms and conditions of this Agreement and restrictions and exceptions set forth in the Software README file, including, but not limited to the Java Technology Restrictions of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license without fees to reproduce and distribute those files specifically identified as redistributable in the Software "README" file ("Redistributables") provided that (i) you distribute the Redistributables complete and unmodified and only bundled as part of Programs, (ii) the Programs add significant and primary functionality to the Redistributables, (iii) you do not distribute additional software intended to supersede any component(s) of the Redistributables (unless otherwise specified in the applicable README file), (iv) you do not remove or alter any proprietary legends or notices contained in or on the Redistributables, (v) you only distribute the Redistributables pursuant to a license agreement that protects Sun's interests consistent with the terms contained in the Agreement, (vi) you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

D. Java Technology Restrictions. You may not create, modify, or change the behavior of, or authorize your licensees to create, modify, or change the behavior of, classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Sun in any naming convention designation.

E. Distribution by Publishers. This section pertains to your distribution of the Software with your printed book or magazine (as those terms are commonly used in the industry) relating to Java technology ("Publication"). Subject to and conditioned upon your compliance with the restrictions and obligations contained in the Agreement, in addition to the license granted in Paragraph I above, Sun hereby grants to you a non-exclusive, nontransferable limited right to reproduce complete and unmodified copies of the Software on electronic media (the "Media") for the sole purpose of inclusion and distribution with your Publication(s), subject to the following terms: (i) You may not distribute the Software on a stand-alone basis; it must be distributed with your Publication(s); (ii) You are responsible for downloading the Software from the applicable Sun web site; (iii) You must refer to the Software as JavaTM 2 Platform Standard Edition Development Kit 5.0; (iv) The Software must be reproduced in its entirety and without any modification whatsoever (including, without limitation, the Binary Code License and Supplemental License Terms accompanying the Software and proprietary rights notices contained in the Software); (v) The Media label shall include the following information: Copyright 2004, Sun Microsystems, Inc. All rights reserved. Use is subject to license terms. Sun, Sun Microsystems, the Sun logo, Solaris, Java, the Java Coffee Cup logo, J2SE, and all trademarks and logos based on Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. This information must be placed on the Media label in such a manner as to only apply to the Sun Software; (vi) You must clearly identify the Software as Sun's product on the Media holder or Media label, and you may not state falsely that Sun is responsible for the Software. You may not include the Sun logo on the Media which is intended to be a trademark or service mark for Software; (vii) You may not include the Sun logo on the Media label or Media holder if you are not a Sun customer; (viii) You may not claim ownership of the Software; (ix) You may not claim that the Software is your creation or that you developed the Software; (x) You may not claim that the Software is your original work; (xi) You may not claim that the Software is your intellectual property; (xii) You may not claim that the Software is your exclusive property; (xiii) You may not claim that the Software is your sole property; (xiv) You may not claim that the Software is your unique property; (xv) You may not claim that the Software is your primary property; (xvi) You may not claim that the Software is your principal property; (xvii) You may not claim that the Software is your most valuable property; (xviii) You may not claim that the Software is your best property; (xix) You may not claim that the Software is your best and most valuable property; (xx) You may not claim that the Software is your best and most valuable intellectual property; (xxi) You may not claim that the Software is your best and most valuable original work; (xxii) You may not claim that the Software is your best and most valuable unique work; (xxiii) You may not claim that the Software is your best and most valuable original intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique work; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxix) You may not claim that the Software is your best and most valuable original unique intellectual property; (xx) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxiv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxv) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvi) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxvii) You may not claim that the Software is your best and most valuable original unique intellectual property; (xxviii) You may not claim that the Software is

Any errors or suggested improvements to this class can be reported as instructed on CoolServlets.com. We hope you enjoy this program... your comments will encourage further development!

This software is distributed under the terms of the BSD License. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither name of CoolServlets.com nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY COOLSERVLETS.COM AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY; WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE."

The following software may be included in this product: Crimson v1.1.1 ; Use of any of this software is governed by the terms of the license below:

The Apache Software License, Version 1.1

Copyright (c) 1999-2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).". Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Crimson" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY; WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE."

=====

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, International Business Machines, Inc., <http://www.ibm.com>. For more information on the Apache Software Foundation, please see <<http://www.apache.org/>>.

The following software may be included in this product: Xalan J2;

Use of any of this software is governed by the terms of the license below:

Apache License

Version 2.0, January 2004

<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the unit of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

The following software may be included in this product: NSIS 1.0; Use of any of this software is governed by the terms of the license below:

Copyright (C) 1999-2000 Nullsoft, Inc.

This software is provided "as-is", without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.Justin Frankel justin@nullsoft.com"

Some Portions licensed from IBM are available at: <http://oss.software.ibm.com/icu4j/>

Portions Copyright Eastman Kodak Company 1992

Lucida is a registered trademark or trademark of Bigelow & Holmes in the U.S. and other countries.

Portions licensed from Taligent, Inc.

The following software may be included in this product:IAIK PKCS Wrapper; Use of any of this software is governed by the terms of the license below:

Copyright (c) 2002 Graz University of Technology. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:

"This product includes software developed by IAIK of Graz University of Technology."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Graz University of Technology" and "IAIK of Graz University of Technology" must not be used to endorse or promote products derived from this software without prior written permission.

5. Products derived from this software may not be called "IAIK PKCS Wrapper", nor may "IAIK" appear in their name, without prior written permission of Graz University of Technology.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE LICENSOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The following software may be included in this product: Document Object Model (DOM) v. Level 3; Use of any of this software is governed by the terms of the license below:

W3Cyy SOFTWARE NOTICE AND LICENSE

<http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications:

1.The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.

2.Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the W3C Software Short Notice should be included (hypertext is preferred, text is permitted) within the body of any redistributed or derivative code.

3.Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

This formulation of W3C's notice and license became active on December 31 2002.

This version removes the copyright ownership notice such that this license can be used with materials other than those owned by the W3C, reflects that ERCIM is now a host of the W3C, includes references to this specific dated version of the license, and removes the ambiguous grant of "use". Otherwise, this version is the same as the previous version and is written so as to preserve the Free Software Foundation's assessment of GPL compatibility and OS's certification under the Open Source Definition. Please see our Copyright FAQ for common questions about using materials from our site, including specific terms and conditions for packages like libwww, Amaya, and Jigsaw. Other questions about this notice can be directed to site-policy@w3.org.

The following software may be included in this product: Xalan, Xerces; Use of any of this software is governed by the terms of the license below:

The Apache Software License, Version 1.1

Copyright (c) 1999-2003 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).". Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, International Business Machines, Inc., <http://www.ibm.com>. For more information on the Apache Software Foundation, please see <<http://www.apache.org/>>.

The following software may be included in this product: W3C XML Conformance Test Suites v. 20020606; Use of any of this software is governed by the terms of the license below:

W3Cyy SOFTWARE NOTICE AND LICENSE

Copyright yy 1994-2002 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. <http://www.w3.org/Consortium/Legal/>

This W3C work (including software, documents, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications, that you make:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.

2. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, a short notice of the following form (hypertext is preferred, text is permitted) should be used within the body of any redistributed or derivative code: "Copyright yy [date-of-software] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. <http://www.w3.org/Consortium/Legal/>"

3. Notice of any changes or modifications to the W3C files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

This formulation of W3C's notice and license became active on August 14 1998 so as to improve compatibility with GPL. This version ensures that W3C software licensing terms are no more restrictive than GPL and consequently W3C software may be distributed in GPL packages. See the older formulation for the policy prior to this date. Please see our Copyright FAQ for common questions about using materials from our site, including specific terms and conditions for packages like libwww, Amaya, and Jigsaw. Other questions about this notice can be directed to site-policy@w3.org.

The following software may be included in this product: W3C XML Schema Test Collection v. 1.16.2; Use of any of this software is governed by the terms of the license below:

W3Cyy DOCUMENT NOTICE AND LICENSE

Copyright yyyy 1994-2002 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

<http://www.w3.org/Consortium/Legal/>

Public documents on the W3C site are provided by the copyright holders under the following license. The software or Document Type Definitions (DTDs) associated with W3C specifications are governed by the Software Notice. By using and/or copying this document, or the W3C document from which this statement is linked, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and distribute the contents of this document, or the W3C document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the document, or portions thereof, that you use:

1. A link or URL to the original W3C document.

2. The pre-existing copyright notice of the original author, or if it doesn't exist, a notice of the form: "Copyright yyyy [date-of-document] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. <http://www.w3.org/Consortium/Legal/>" (Hypertext is preferred, but a textual representation is permitted.)

3. If it exists, the STATUS of the W3C document.

When space permits, inclusion of the full text of this NOTICE should be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of W3C documents is granted pursuant to this license. However, if additional requirements (documented in the Copyright FAQ) are satisfied, the right to create modifications or derivatives is sometimes granted by the W3C to individuals complying with those requirements.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

=====

This formulation of W3C's notice and license became active on April 05 1999 so as to account for the treatment of DTDs, schema's and bindings. See the older formulation for the policy prior to this date. Please see our Copyright FAQ for common questions about using materials from our site, including specific terms and conditions for packages like libwww, Amaya, and Jigsaw. Other questions about this notice can be directed to site-policy@w3.org.

The following software may be included in this product: Mesa 3-D graphics library v. 5; Use of any of this software is governed by the terms of the license below:

core Mesa code include/GL/gl.h Brian Paul Mesa

GLX driver include/GL/glx.h Brian Paul Mesa

Ext registry include/GL/glxext.h SGI SGI Free B

include/GL/glxnext.h

Mesa license:

The Mesa distribution consists of several components. Different copyrights and licenses apply to different components. For example, GLUT is copyrighted by Mark Kilgard, some demo programs are copyrighted by SGI, some of the Mesa device drivers are copyrighted by their authors. See below for a list of Mesa's components and the copyright/license for each.

The core Mesa library is licensed according to the terms of the XFree86 copyright (an MIT-style license). This allows integration with the XFree86/DRI project. Unless otherwise stated, the Mesa source code and documentation is licensed as follows:

Copyright (C) 1999-2003 Brian Paul. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL BRIAN PAUL BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SGI Free Software License B:

, or is under common control with Recipient. For purposes of this definition, "control" of an entity means (a) the power, direct or indirect, to direct or manage such entity, or (b) ownership of fifty percent (50%) or more of the outstanding shares or beneficial ownership of such entity.

1.12."Recipient Patents" means patent claims licensable by a Recipient that are infringed by the use or sale of Original Code or any Modifications provided by SGI, or any combination thereof.

1.13."SGI" means Silicon Graphics, Inc.

1.14."SGI Patents" means patent claims licensable by SGI other than the Licensed Patents.

2.License Grant and Restrictions.

2.1.SGI License Grant. Subject to the terms of this License and any third party intellectual property claims, for the duration of intellectual property protections inherent in the Original Code, SGI hereby grants Recipient a worldwide, royalty-free, non-exclusive license, to do the following: (i) under copyrights licensable by SGI, to reproduce, distribute, create derivative

The following software may be included in this product: Byte Code Engineering Library (BCEL) v. 5; Use of any of this software is governed by the terms of the license below:

Apache Software License

The Apache Software License, Version 1.1

Copyright (c) 2001 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).". Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" and "Apache BCEL" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", "Apache BCEL", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <<http://www.apache.org/>>.

The following software may be included in this product: Regexp, Regular Expression Package v. 1.2; Use of any of this software is governed by the terms of the license below:

The Apache Software License, Version 1.1

Copyright (c) 2001 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:

"This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).". Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" and "Apache Turbine", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

5. Products derived from this software may not be called "Apache", "Apache Turbine", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org>.

The following software may be included in this product: JLex: A Lexical Analyzer Generator for Java v. 1.2.5; Use of any of this software is governed by the terms of the license below:

JLEX COPYRIGHT NOTICE, LICENSE AND DISCLAIMER.

Copyright 1996-2003 by Elliot Joel Berk and C. Scott Ananian.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of the authors or their employers not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

The authors and their employers disclaim all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall the authors or their employers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

Java is a trademark of Sun Microsystems, Inc. References to the Java programming language in relation to JLex are not meant to imply that Sun endorses this product.

The following software may be included in this product: SAX v. 2.0.1; Use of any of this software is governed by the terms of the license below:

Copyright Status

SAX is free!

In fact, it's not possible to own a license to SAX, since it's been placed in the public domain.

No Warranty

Because SAX is released to the public domain, there is no warranty for the design or for the software implementation, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide SAX "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of SAX is with you.

Should SAX prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute SAX, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use SAX (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the SAX to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

Copyright Disclaimers

This page includes statements to that effect by David Megginson, who would have been able to claim copyright for the original work.

SAX 1.0

Version 1.0 of the Simple API for XML (SAX), created collectively by the membership of the XML-DEV mailing list, is hereby released into the public domain.

No one owns SAX: you may use it freely in both commercial and non-commercial applications, bundle it with your software distribution, include it on a CD-ROM, list the source code in a book, mirror the documentation at your own web site, or use it in any other way you see fit.

David Megginson, sax@megginson.com 1998-05-11

SAX 2.0

I hereby abandon any property rights to SAX 2.0 (the Simple API for XML), and release all of the SAX 2.0 source code, compiled code, and documentation contained in this distribution into the Public Domain. SAX comes with NO WARRANTY or guarantee of fitness for any purpose.

David Megginson, david@megginson.com

2000-05-05

The following software may be included in this product: Cryptix;

Use of any of this software is governed by the terms of the license below:

Cryptix General License

Copyright © 1995-2003 The Cryptix Foundation Limited. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1.Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

2.Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE CRYPTIX FOUNDATION LIMITED AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE CRYPTIX FOUNDATION LIMITED OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

EXERPT FROM JavaTM 2 Platform Standard Edition Development Kit 5.0 README

You can freely redistribute the J2SE Runtime Environment with your application, according to the terms of the Runtime Environment's license. Once you have developed your application using the JDK, you can ship it with the Runtime Environment so your end-users will have a Java platform on which to run your software.

Redistribution

NOTE - The license for this software does not allow the redistribution of beta and other pre-release versions.

Subject to the terms and conditions of the Software License Agreement and the obligations, restrictions, and exceptions set forth below, You may reproduce and distribute the Software (and also portions of Software identified below as Redistributable), provided that:

you distribute the Software complete and unmodified and only bundled as part of Your applets and applications ("Programs"),

your Programs add significant and primary functionality to the Software,

your Programs are only intended to run on Java-enabled general purpose desktop computers and servers,

you distribute Software for the sole purpose of running your Programs,

you do not distribute additional software intended to replace any component(s) of the Software,

you do not remove or alter any proprietary legends or notices contained in or on the Software,

you only distribute the Software subject to a license agreement that protects Sun's interests consistent with the terms contained in this Agreement, and

you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

The term "vendors" used here refers to licensees, developers, and independent software vendors (ISVs) who license and distribute the J2SE Development Kit with their programs.

Vendors must follow the terms of the J2SE Development Kit Binary Code License agreement.

Required vs. Optional Files

The files that make up the J2SE Development Kit are divided into two categories: required and optional. Optional files may be excluded from redistributions of the JDK at the vendor's discretion.

The following section contains a list of the files and directories that may optionally be omitted from redistributions of the JDK. All files not in these lists of optional files must be included in redistributions of the JDK.

Optional Files and Directories

The following files may be optionally excluded from redistributions. These files are located in the jdk1.5.0_<version> directory, where <version> is the update version number. Solaris and Linux filenames and separators are shown. Windows executables have the ".exe" suffix. Corresponding files with _g in name can also be excluded.

jre/lib/charsets.jar
Character conversion classes
jre/lib/ext/
sunjce_provider.jar - the SunJCE provider for Java Cryptography APIs
localizedata.jar - contains many of the resources needed for non US English locales
ldapsec.jar - contains security features supported by the LDAP service provider
dnsns.jar - for the InetAddress wrapper of JNDI DNS provider
bin/rmid and jre/bin/rmid
Java RMI Activation System Daemon
bin/rminregistry and jre/bin/rminregistry
Java Remote Object Registry
bin/nameserv and jre/bin/nameserv
Java IDL, Name Server
bin/keytool and jre/bin/keytool
Key and Certificate Management Tool
bin/kinit and jre/bin/kinit
Used to obtain and cache Kerberos ticket-granting tickets
bin/klist and jre/bin/klist
Kerberos display entries in credentials cache and keytab
bin/ktab and jre/bin/ktab
Kerberos key table manager
bin/policytool and jre/bin/policytool
Policy File Creation and Management Tool
bin/orbd and jre/bin/orbd
Object Request Broker Daemon
bin/servertool and jre/bin/servertool
Java IDL Server Tool
bin/javaws, jre/bin/javaws, jre/lib/javaws/ and jre/lib/javaws.jar
Java Web Start
src.zip
Archive of source files
Redistributable JDK File

The limited set of files from the JDK listed below may be included in vendor redistributions of the J2SE Runtime Environment. They cannot be redistributed separately, and must accompany a JRE distribution. All paths are relative to the top-level directory of the JDK.

jre/lib/cmmn/PYCC.pf
Color profile. This file is required only if one wishes to convert between the PYCC color space and another color space.
All.ttf files in the jre/lib/fonts directory.
Note that the LucidaSansRegular.ttf font is already contained in the J2SE Runtime Environment, so there is no need to bring that file over from the JDK.
jre/lib/audio/soundbank.gm

This MIDI soundbank is present in the JDK, but it has been removed from the J2SE Runtime Environment in order to reduce the size of the Runtime Environment's download bundle. However, a soundbank file is necessary for MIDI playback, and therefore the JDK's soundbank.gm file may be included in redistributions of the Runtime Environment at the vendor's discretion. Several versions of enhanced MIDI soundbanks are available from the Java Sound web site: <http://java.sun.com/products/java-media/sound/>. These alternative soundbanks may be included in redistributions of the J2SE Runtime Environment.

The javac bytecode compiler, consisting of the following files:
bin/java [Solaris(TM) Operating System and Linux]
bin/sparcv9/javac [Solaris Operating System (SPARC(R) Platform Edition)]
bin/amd64/javac [Solaris Operating System (AMD)]
bin/java.exe [Microsoft Windows]
lib/tools.jar [All platforms]

The Annotation Processing Tool, consisting of the following files:
bin/apt [Solaris(TM) Operating System and Linux]
bin/sparcv9/apt [Solaris Operating System (SPARC(R) Platform Edition)]
bin/amd64/apt [Solaris Operating System (AMD)]
bin/apt.exe [Microsoft Windows]
jre/bin/server/

On Microsoft Windows platforms, the JDK includes both the Java HotSpot Server VM and Java HotSpot Client VM. However, the J2SE Runtime Environment for Microsoft Windows platforms includes only the Java HotSpot Client VM. Those wishing to use the Java HotSpot Server VM with the J2SE Runtime Environment may copy the JDK's jre/bin/server folder to a bin/server directory in the J2SE Runtime Environment. Software vendors may redistribute the Java HotSpot Server VM with their redistributions of the J2SE Runtime Environment.

Unlimited Strength Java Cryptography Extension

Due to import control restrictions for some countries, the Java Cryptography Extension (JCE) policy files shipped with the J2SE Development Kit and the J2SE Runtime Environment allow strong but limited cryptography to be used. These files are located at
<java-home>/lib/security/local_policy.jar
<java-home>/lib/security/US_export_policy.jar
where <java-home> is the directory of the JDK or the top-level directory of the J2SE Runtime Environment.

An unlimited strength version of these files indicating no restrictions on cryptographic strengths is available on the JDK web site for those living in eligible countries. Those living in eligible countries may download the unlimited strength version and replace the strong cryptography jar files with the unlimited strength files.

jconsole
jconsole.jar
jconsole may be redistributed outside the JDK but only with Sun's JRE.

Endorsed Standards Override Mechanism

An endorsed standard is a Java API defined through a standards process other than the Java Community ProcessSM (JCPSM). Because endorsed standards are defined outside the JCP, it is anticipated that such standards will be revised between releases of the Java 2 Platform. In order to take advantage of new revisions to endorsed standards, developers and software vendors may use the Endorsed Standards Override Mechanism to provide newer versions of an endorsed standard than those included in the Java 2 Platform as released by Sun Microsystems.

For more information on the Endorsed Standards Override Mechanism, including the list of platform packages that it may be used to override, see
<http://java.sun.com/j2se/1.5.0/docs/guide/standards/>

Classes in the packages listed on that web page may be replaced only by classes implementing a more recent version of the API as defined by the appropriate standards body.

In addition to the packages listed in the document at the above URL, which are part of the Java 2 Platform Standard Edition (J2SE(TM)) specification, redistributors of Sun's J2SE Reference Implementation are allowed to override classes whose sole purpose is to implement the functionality provided by public APIs defined in these Endorsed Standards packages. Redistributors may also override classes in the org.w3c.dom.* packages, or other classes whose sole purpose is to implement these APIs.

The cacerts Certificates File

Root CA certificates may be added to or removed from the J2SE certificate file located at <java-home>/lib/security/cacerts. For more information, see the cacerts Certificates File section in the keytool documentation.

Web Pages

For additional information, refer to these Sun Microsystems pages on the World Wide Web:
<http://java.sun.com/>
The Java Software web site, with the latest information on Java technology, product information, news, and features.
<http://java.sun.com/docs>

Java Platform Documentation provides access to white papers, the Java Tutorial and other documents.
<http://developer.java.sun.com>

Developer Services web site. (Free registration required.) Additional technical information, news, and features; user forums; support information, and much more.
<http://java.sun.com/products/>

Java Technology Products & API

The J2SE Development Kit is a product of Sun MicrosystemsTM, Inc.
Copyright 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A.
All rights reserved.

JDOM

Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <request_AT_jdom_DOT_org>.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management <request_AT_jdom_DOT_org>.

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following:

"This product includes software developed by the JDOM Project (<http://www.jdom.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.jdom.org/images/logos>.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter_AT_jdom_DOT_org> and Brett McLaughlin <brett_AT_jdom_DOT_org>. For more information on the JDOM Project, please see <<http://www.jdom.org/>>.

Krypto

Copyright (c) 1997 Stanford University

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notices and this permission notice appear in all copies of the software and related documentation.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL STANFORD BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (C) 1995-1997 Eric Young (eay@mincom.oz.au)

All rights reserved.

This package is an SSL implementation written by Eric Young (eay@mincom.oz.au). The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are abeared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, Ihash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@mincom.oz.au).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

"This product includes cryptographic software written by Eric Young (eay@mincom.oz.au)"

The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@mincom.oz.au)"

"This product includes software written by Tim Hudson (tjh@mincom.oz.au)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publicilly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence].

OpenLDAP

Public License for 2.3.34

The OpenLDAP Public License

Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices.
2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and
3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distribute verbatim copies of this document is granted.

OpenSSL

Licence

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL, please contact openssl-core@openssl.org.

OpenSSL License

Copyright (c) 1998-2007 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment:

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following acknowledgment:

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OPENSSL PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)

All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are abeared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, Ihash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

"This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)"

The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License.]

Oracle

Oracle Instant client
End user license agreement ("Agreement")

MatrixOne Inc., ("MatrixOne") as licensor, has been given the right by Oracle Corporation (Oracle") to distribute the Oracle Instant Client software ("Programs") to you, an end user. Each end user hereby agrees: (1) to restrict its use of the Programs to its internal business operations; (2) that it is prohibited from (a) reselling, giving, or transferring the Programs or an interest in them to another individual or entity (and if it grants a security interest in the Programs, the secured party has no right to use or transfer the Programs); (b) making the Programs available in any manner to any third party for use in the third party's business operations (unless such use is expressly permitted for the specific program license or materials listed in the services required); and (3) that title to the Programs does not pass to the end user or any other party; (4) that reverse engineering is prohibited (unless required by law for the use of the Programs); (5) directly or indirectly, to (i) copy or reproduce any portion of the Programs, including any documentation, or (ii) to make any derivative work based on the Programs; (6) to use the Programs for commercial purposes; (7) that, to the extent prohibited by applicable law, liability of Oracle and MatrixOne for any damages, whether direct, indirect, incidental, or consequential, arising from the use of the Programs is disclaimed; (8) at the termination of the Agreement, to discontinue use and destroy or return to MatrixOne all copies of the Programs and documentation; (9) not to publish any results of benchmark tests run on the Programs; (10) to comply fully with all relevant export laws and regulations of the United States and other applicable export and import laws to assure that neither the Programs, nor any direct product thereof, are exported, directly or indirectly, in violation of applicable laws and are not used for any purpose prohibited by these laws including, without limitation, nuclear, chemical or biological weapons proliferation; (11) that Oracle is not required to perform any obligations or incur any liability not previously agreed to; (12) to permit MatrixOne to audit its use of the Programs or to assign such audit right to Oracle; (13) that Oracle is a third party beneficiary of this end user license agreement; (14) that the application of the Uniform Computer Information Transactions Act is excluded.

Disclaimer of Warranty and Exclusive Remedies

THE PROGRAMS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. MATRIXONE AND ORACLE FURTHER DISCLAIM ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT.

IN NO EVENT SHALL MATRIXONE OR ORACLE BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR DATA USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, EVEN IF MATRIXONE OR ORACLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. MATRIXONE'S AND ORACLE'S ENTIRE LIABILITY FOR DAMAGES HEREUNDER SHALL IN NO EVENT EXCEED ONE THOUSAND DOLLARS (U.S. \$1,000).

No Technical Support

Oracle and MatrixOne technical support organizations will not provide technical support, phone support, or updates to end users for the Programs licensed under this agreement.

Restricted Rights

For United States government end users, the Programs, including documentation, shall be considered commercial computer software and the following applies:

NOTICE OF RESTRICTED RIGHTS

"Programs delivered subject to the DOD FAR Supplement are 'commercial computer software' and use, duplication, and disclosure of the programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, programs delivered subject to the Federal Acquisition Regulations are 'restricted computer software' and use, duplication, and disclosure of the programs, including documentation, shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065."

End of Agreement

The end user may terminate this Agreement by destroying all copies of the Programs. MatrixOne and Oracle each have the right to terminate the end user's right to use the Programs if the end user fails to comply with any of the terms of this Agreement, in which case the end user shall destroy all copies of the Programs.

Relationship Between the Parties

The relationship between the end user and MatrixOne and Oracle is that the end user is licensee, MatrixOne is distributor/licensor and Oracle is licensor. No party will represent that it has any authority to assume or create any obligation, express or implied, on behalf of any other party, nor to represent the other party as agent, employee, franchisee, or in any other capacity. Nothing in this Agreement shall be construed to limit any party's right to independently develop or distribute software that is functionally similar to the other party's products, so long as proprietary information of the other party is not included in such software.

Open Source

"Open Source" software - software available without charge for use, modification and distribution - is often licensed under terms that require the user to make the user's modifications to the Open Source software or any software that the user 'combines' with the Open Source software freely available in source code form. If you as end user use Open Source software in conjunction with the Programs, you must ensure that your use does not: (i) create, or purport to create, obligations of MatrixOne or Oracle with respect to the Oracle Programs; or (ii) grant, or purport to grant, to any third party any rights to or immunities under intellectual property or proprietary rights in the Oracle Programs. For example, you may not develop a software program using an Oracle Program and an Open Source program where such use results in a program file(s) that contains code from both the Oracle Program and the Open Source program (including without limitation libraries) if the Open Source program is licensed under a license that requires any "modifications" be made freely available. You also may not combine the Oracle Program with programs licensed under the GNU General Public License ("GPL") in any manner that could cause, or could be interpreted or asserted to cause, the Oracle Program or any modifications thereto to become subject to the terms of the GPL.

SSLutils

The Apache Software License, Version 1.1

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).". Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "SOAP" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. Sun RPC is provided with no support and without any obligation on the part of Sun Microsystems, Inc. to assist in its use, correction, modification or enhancement.

SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY SUN RPC OR ANY PART THEREOF.

In no event will Sun Microsystems, Inc. be liable for any lost revenue or profits or other special, indirect and consequential damages, even if Sun has been advised of the possibility of such damages.

Sun Microsystems, Inc.

2550 Garcia Avenue

Mountain View, California 94043

Tcl

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF; EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

Xalan

[under Apache License, Version 2.0 above]

Xerces

[under Apache License, Version 2.0 above]

Xerces2

[under Apache License, Version 2.0 above]

Overview

Background Details for Developing Applications

- [About the Product Architecture](#)
- [Evolving Architecture](#)
- [About JSPs](#)
- [About Java Program Objects \(JPOs\)](#)
- [About JavaBeans](#)
- [Property Files, Language Strings, and Settings](#)
- [Studio Customization Toolkit Overview](#)
- [Setting up Java Packages](#)
- [Business Process Services Overview](#)

UI Basics

[Dynamic User Interface](#)

- [About the Dynamic User Interface](#)
- [Naming Conventions for UI Administrative Objects](#)
- [Using Macros and Expressions in Dynamic UI Components](#)
- [Internationalizing Dynamic UI Components](#)

[User Access to UI Components](#)

- [Access Checks for Individual Components](#)
- [Search Page Access](#)
- [Access Checks for Messages](#)
- [Access Parameters and Settings](#)
- [Sample JPO for Controlling Access](#)

[Configuring Styles](#)

[Main Navigator Page](#)

- [About the Content Page for emxNavigator.jsp](#)
- [Configuring the Main Navigator Page](#)
- [Configuring the Default Home Page for the Content Frame](#)
- [URL Parameters Accepted by emxNavigator.jsp](#)

[Adding a Custom Application](#)

Migration to the New UI

- [About the New User Interface \(UI\)](#)
- [About the FS Page Component](#)
- [Updating Framesets and DOCTYPE References](#)
- [Converting JavaScript Frame References](#)
- [Converting Pages to the New UI Framework](#)
- [Migrating to the New UI - Optional Steps](#)
- [Migrating to Slide-in Frames](#)
- [Removing Inline Style Definitions and Invalid HTML Elements](#)
- [Enabling Automatic Type Ahead](#)
- [Updating PowerView Channel Heights](#)
- [Table-to-Structure Browser Conversion](#)

Extending Applications

[Java Coding Hints](#)

[Using JPOs](#)

- [About JPOs](#)
- [Simple JPO Example](#)
- [Accessing the Studio Customization Toolkit](#)
- [Passing Objects into a JPO Method](#)

[Round-trip Engineering](#)

- [Compiling for Round-trip Engineering](#)
- [Editing to Support Round-trips](#)
- [Extracting to Support Round-trips](#)
- [Inserting to Support Round-Trips](#)
- [Inserting and Extracting to Support Round-Trips](#)
- [Debugging JPOs for Round-trip Engineering](#)
- [Java and exec Commands](#)

[Using Servlets](#)

- [Servlet Processing](#)
- [Accessing the Servlet](#)

[JavaServer Pages](#)

- [About Mapping Files and JavaBeans](#)
- [Multiple Jar File Architecture](#)
- [Business Process Services Programming](#)
- [Naming Conventions](#)
- [Extending JPOs](#)
 - [About Extending JPOs](#)
 - [Extending Beans](#)
 - [Extending JSPs](#)
- [About Writing JSPs](#)

Configurable Pages and Commands

[Configurable Preference Pages](#)

- [About Preferences](#)
- [Adding a Preference Page](#)
- [Implementing a Currency Conversion](#)

- [Defining a Conversion Rate Preference](#)
- [Configurable History Page](#)
 - [About Configurable History Pages](#)
 - [Parameters Used by emxHistory.jsp](#)
 - [Including Configurable History in a Custom Application](#)
- [Configurable Type Chooser](#)
 - [About the Configurable Type Chooser](#)
 - [Parameters Used by the Type Chooser](#)
 - [Calling the Configurable Type Chooser from a Page](#)
- [Date Chooser](#)
- [Lifecycle Page](#)
- [Logout Command](#)
- [Change Password Command](#)
- [Page History Page and Command](#)
- [Quick File Access Details Table](#)
- [Context-Sensitive Help](#)
 - [About Context-Sensitive Help](#)
 - [Implementing Context-Sensitive Help](#)
 - [Implementing Context-Sensitive Help for XML Source](#)

Web Services

- [About Web Services](#)
- [Developing and Deploying a Web Service](#)
- [Web Service Examples](#)

Triggers

- [About Event Triggers](#)
- [How Rules are Automated](#)
- [Trigger Scenarios](#)
- [Designing Triggers](#)
 - [Supported Trigger Events](#)
 - [Types of Triggers](#)
 - [Multi-trigger Events](#)
 - [Transaction Boundaries](#)
 - [Notification API and Triggers](#)
 - [Modifying Inputs Trigger Programs](#)
- [Trigger Programs](#)
 - [Adding a Trigger for a Trigger Event](#)
 - [Validate Query Trigger](#)
 - [Recursion Detection Modes and Limits](#)
 - [Enabling/Disabling Triggers](#)
- [Utility Trigger Programs Reference](#)
 - [Previous Revision Promotion](#)
 - [Required Connection Check](#)
 - [Required File Check](#)
 - [Check Relative State](#)
 - [Valid Revision Sequence](#)
 - [Relative Float Action](#)
 - [Set Originator Attribute](#)

Charts

- [Working With Charts](#)
- [Adding a Chart to a Page](#)
- [URL Parameters for emxChart.jsp](#)

Reports

- [About Reports](#)
- [Installing BusinessObjects XI Release 2](#)
- [Installing Crystal Reports Professional](#)
- [Verifying the Configuration](#)
- [Converting Crystal Reports 10](#)
- [Running Sample Reports](#)
- [Modifying the System Properties for Reports](#)
- [Implementing the Crystal Reports Interface](#)
- [Creating a Crystal Report File with the MatrixJPO Data Source](#)
- [Creating a JPO for the RPT File](#)
- [Creating a Crystal Report File with the MatrixInquiry Data Source](#)
- [Creating an Inquiry To Be Used by the RPT File](#)
- [Sample Reports and JPOs](#)

Menus and Toolbars

- [Global Toolbar Menu](#)
- [Menus](#)
 - [About Building Menus](#)
 - [Building a Menu](#)
 - [Global Search Menu](#)
 - [About the Global Search Menu](#)
 - [Parameters and Settings for AEFGlobalSearch Menu](#)
 - [Parameters and Settings for a Command](#)
- [Right-click Menus](#)
 - [About Right-Click Menus](#)

[Dynamic Commands in a Right-Click Menu](#)

[Toolbars](#)

[About Toolbars](#)

[Input Controls](#)

[Input Controls](#)

[Textbox Input Control](#)

[Textbox Input Control Examples](#)

[Combobox Input Control](#)

[Checkbox Input Control](#)

[Submit Button Input Control](#)

[Adding Input Controls to the Toolbar](#)

[Configurable Toolbar](#)

[Building a Configurable Toolbar](#)

[Toolbar Filters for the Structure Browser](#)

[Parameters for Toolbar Menu Objects](#)

[Settings for Toolbar Menu Objects](#)

[Parameters for Toolbar Link Command Objects](#)

[Settings for Toolbar Link Command Objects](#)

[Implementing a Toolbar in a JSP](#)

Searches

[Configurable Search Webform](#)

[About the Configurable Search Webform](#)

[Settings for emxFormEditDisplay.jsp](#)

[URL Parameters Accepted by emxFormEditDisplay.jsp](#)

[Full-Text Search](#)

[About emxFullSearch.jsp \(Full-Text Search\)](#)

[Initial Search Criteria](#)

[Full-Text Search Results Toolbar](#)

[Full-Text Search from a Context Page](#)

[Dynamically-Added Columns to the Results](#)

[Search Results Sorting Options](#)

[Form-Based Search](#)

[Criteria For Real-Time Searches](#)

[Field Choosers for Full-Text Search](#)

[Full-Text Search Page Used as a Chooser](#)

[URL Parameters Passed to emxFullSearch.jsp](#)

Navigation Trees

[About Navigation Trees](#)

[Structure Navigator](#)

[Definition of a Navigation Tree](#)

[Sub-trees Configured as Categories](#)

[Dynamic Categories for a Tree](#)

[Structure Tree for an Object](#)

[Navigation Tree with Revision Filters](#)

[Navigation Trees Linked to an Application](#)

[Building a Navigation Tree](#)

[Tree JavaScript APIs for Custom JSPs](#)

[About the Tree JavaScript API for Custom JSPs](#)

[Obtaining the Details Tree Object](#)

[Obtaining the Structure Tree Object](#)

[Getting the Selected Node](#)

[Getting a Node](#)

[Getting a Node's nodeID](#)

[Getting a Node's Parent Node](#)

[Getting Current Tree Root](#)

[Getting Tree Current Root object ID](#)

[Getting Tree Original Root object ID](#)

[Setting the Selected Node](#)

[Changing the Name of All Nodes Representing a Business Object](#)

[Changing the Object ID of All Nodes Representing a Business Object](#)

[Determining if an Object Exists in the Tree](#)

[Deleting All Nodes Representing a Business Object](#)

[Removing a Single Node](#)

[Inserting a New Node into Details Tree](#)

[Inserting a New Node into Structure Tree](#)

[Deleting Nodes in Both Details and Structure Trees](#)

[Deprecated Methods \(Navigation Tree\)](#)

[Parameters for Tree Menu Objects](#)

[Settings for Tree Menu Objects](#)

[Parameters for Tree Category Command Objects](#)

[Settings for Tree Category Command Objects](#)

[URL Parameters Accepted by emxTree.jsp](#)

[About the Default Tree](#)

[Specific Trees for an Object Type](#)

[About the Icon for a Type's Tree](#)

[Dynamic Expand for Trees](#)

[About Dynamic Expand for Tree Categories](#)

[Requirements for a Dynamic Expand JPO](#)

[Examples of Dynamic Expand Inquiries](#)

Guidelines for Writing Structure Tree JPO

Accessing Applications Externally

About External Access to Applications

Content of the Navigator Page

Location of the Navigator Page

Forms

About Forms (emxForm.jsp)

Date/Time Fields in Forms and Tables

HTML Components for a Form

Building a Form Page

Form Pre/Post/Cancel Processing

About Form Processing

Pre-Process URL for an Editable Form Page

Pre-Process JPO for an Editable Form Page

Cancel Process URL for an Editable Form Page

Cancel Process JPO for an Editable Form Page

Post Process URL for an Editable Form Page

Post Process JPO for an Editable Form Page

Passing Field Map Information to a Custom JPO

Configurable Form View of Multiple Rows and Columns with programHTMLOutput

Additional URL Parameters for Forms

Parameters for Web Form Objects

Settings for Fields in Web Form Objects

URL Parameters Accepted by emxForm.jsp

Actions Menu in a Configurable Form

Filter Toolbar for a Configurable Form

Design Considerations for Form Fields

Form Fields

Field Values as Select Expressions

Field Values Obtained from a Program

Field Values as a Hyperlink and Type Icon

Field Values with Hyperlinked Data Using an Alternate OID and Alternate Type Icon

Field Values with an Additional Icon

Field Values as Hyperlinked Image

Field for Attribute with Choices in Combo Box

Field for Attribute with Choices in Radio Buttons

Fields with Checkboxes

Field for Attribute with Choices as Checkboxes

Field Value with Dates

Field Values as Units of Measure

Form Fields as Arithmetic Expression

Fields as Dynamic Textareas

Field as Section Header and Separator

Fields that are Grouped

Fields Arranged in a Table

Field that Embeds a Configurable Table

Field with Popup Range Helper

Fields with Dynamic Attributes

Implementing Range Helpers for Choosers or Custom Pages

Validating Form Field Data

Committing Changes Made on Edit Form

Generic Create Form

About the Generic Create Form

Results of the Create Page

Create Page Toolbar

Defining the Basic Attributes

Additional Fields on the Create Page

Grouping Fields on the Create Form

Parameters Supported by emxCreate.jsp

URL Parameters Accepted by emxCreate.jsp

Settings Supported by emxCreate.jsp

Create Form Code Samples

JPO Interface for Form Fields

About Configurable Form Support for JPOs

JPO Method Signature and Input Argument

JPO for Getting Field Range Values (Choices)

JPO for Getting Field Values

Update Methods In JPO Programs

Sample JPO for Getting Field Values

Sample JPO for Web Form with Custom Combobox

Tables

About Tables

Building a Table

Editable Tables

Editable Table Configuration

Invoking a Table Directly in Edit Mode

Toolbars for an Editable Table

Using Type Ahead in a Table Column

Manual Entry for Combo Boxes

- [Edit Access for Selected Table Objects](#)
- [Dynamic URLs and mxLinks for a Column](#)
- [Adding an Editable Table to an Application](#)
- [Pre/Post/Cancel Processing of an Editable Table](#)
 - [Pre/Post/Cancel Processing of an Editable Table](#)
 - [Preprocess URL for an Editable Table](#)
 - [Preprocess JPO for an Editable Table](#)
 - [Cancel Process URL for an Editable Table](#)
 - [Cancel Process JPO for an Editable Table](#)
 - [Post Process URL for an Editable Table](#)
 - [Post Process JPO for an Editable Table](#)
- [Parameters for Table Objects](#)
- [Refreshing the Table After Adding, Editing, or Deleting Objects](#)
- [Object or Relationship ID of a Selected Table Item](#)
- [Table Columns](#)
 - [Icons in Column Headings](#)
 - [Settings for Table Column Objects](#)
 - [Table Column Data Definition](#)
 - [Column Values as Select Expressions](#)
 - [Column Values as Program Output](#)
 - [Column Values as Check Boxes](#)
 - [Column Values as Images](#)
 - [Column Values as Icons](#)
 - [Column Values Using Alternate OID in href and Select Expression](#)
 - [Column Values Using Alternate OID and Type Icon in href with Select Expression](#)
 - [Column Values Including Additional Icons](#)
 - [Column Values as Units of Measure](#)
 - [Column Values as Quick File Access](#)
 - [Grouped Columns and Column Separator](#)
 - [Column Values as Special Information from Business Object](#)
 - [Editable Table Columns](#)
 - [Sort Orders for Table Columns](#)
 - [Group By Column](#)
 - [Dynamic Columns](#)
 - [Table Column Calculations](#)
 - [Units of Meaure for Columns](#)
 - [Improving Performance of Table Columns](#)
- [Additional URL Parameters for Tables](#)
- [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#)
- [JPO Guidelines for Tables](#)
 - [JPO Guidelines for Getting Object List](#)
 - [Sample JPO for Getting List of Objects](#)
 - [JPO Guidelines for Updating Column Values](#)
- [Custom Filter for a Table](#)
 - [Implementing a Custom Filter](#)
 - [Sample JSPs for Custom Filtering](#)
- [Sorting Programs for Tables](#)
 - [Guidelines for Custom Sort JPO](#)
 - [Sample Sorting Code](#)
 - [BPS Custom Sorting Programs](#)
- [Parameters and Settings for Inquiry Objects](#)

Structure Browser

- [About the Structure Browser](#)
- [Structure Browser View Mode](#)
 - [Structure Browser - View Mode](#)
 - [Structure Browser Header](#)
 - [Structure Browser Column Headings](#)
 - [Structure Browser View - Table Display](#)
 - [Parts of the Table in a Structure Browser](#)
 - [Root Node\(s\) Label](#)
 - [Check Box](#)
 - [Table Display](#)
 - [Merged Cells](#)
 - [Column Access Control](#)
 - [Column Data Display](#)
 - [Column Sort](#)
 - [Grouped Columns and a Column Separator](#)
 - [Custom Cell Styles](#)
 - [Freeze Pane](#)
 - [Expand Objects using Relationship](#)
 - [Expand Objects using JPO for ENOVIA Objects](#)
 - [Expand using JPO for Non-object Based Browsers](#)
 - [Expand Parameters and Precedence](#)
 - [Custom Expand Filter](#)
 - [Example JPO Code For Custom Expand Filter](#)
 - [Filters](#)
 - [Manipulate Selected Rows](#)
- [Structure Browser Edit Mode](#)
 - [About Edit Mode](#)
 - [Custom Logic for Connections in the Structure Browser](#)

Validating Edit Operations
Sample JPO Program for connectionProgram
Pre/Post/Cancel Processing for Editable Structure Browser
About Processing the Structure Browse Edit View
Pre Process URL for an Editable Structure Browser
Pre Process JPO for an Editable Structure Browser
Cancel Process URL for an Editable Structure Browser
Cancel Process JPO for an Editable Structure Browser
Post Process URL for an Editable Structure Browser
Post Process JPO for an Editable Structure Browser
Mass Update Toolbar
In-cell Editing
Markup Interface
Editable Columns
Adding Rows for New or Existing Objects to the Structure
Validating Cell Values
Custom Client-side Validation
JavaScript APIs Available for Validation Methods
XHTML Standard
Structure Browser in Flat Table Mode
Filter for a Structure Browser Page
Calculations for the Structure Browser
About Column Calculations
Calculation Settings
Decimal Precision
Custom Calculations
Arithmetic Calculations
Structure Compare
Default Structure Compare Dialog Box
Column Settings for Structure Compare
Custom Structure Compare Dialog Box
Structure Compare Report
URL Parameters Accepted by emxStructureCompare.jsp and emxStructureCompareReport.jsp
Streaming Query and Expand
About Streaming Query and Expand
Enabling Streaming Query in a Custom Application
Enabling Streaming Expand in a Custom Application
Additional URL Parameters for Structure Browsers
URL Parameters Accepted by emxIndentedTable.jsp
Settings for Columns in a Structure Browser
Parameters for Structure Browser

Subscriptions

About Subscriptions
Subscription Options Dialogs
Pushed Subscriptions
Subscriptions for Administrative Types
Subscriptions for Relationships
Notifications for Subscriptions
About Notifications for Subscriptions
Sample XML Template for Body Text
Sample XML Template for Body HTML
Sample JPO Code for Notification Attributes

Email Listener
About the Email Listener
The mailListener.xml File
Preprocessing Email Messages
Invoking the Email Listener JPO
Troubleshooting the Email Listener

Automatic Type Ahead

About Automatic Type Ahead
Defining Automatic Type Ahead for a Person Field/Column
Defining Automatic Type Ahead for an Organization
Defining Automatic Type Ahead Using Advanced Search

DesignSync File Access

DesignSync File Access (DSFA) Overview
DSFA Operations
About DSFA Operations DSFA
Connect to a File in DesignSync
Connect to a Folder in DesignSync
Connect to a Module in DesignSync
Copy Files to Other DesignSync Stores
Modify a Connection
Disconnect from a File, Folder, or Module in DesignSync
Delete a File or Module from a DesignSync Store
FCS and DesignSync
Check in DesignSync Files
Check in DesignSync Folders
Check in Physical Data to DesignSync Modules

- [Check Out DesignSync Files](#)
- [Check Out DesignSync Folders](#)
- [Check out a DesignSync Module](#)
- [Write Two-Part Checkin of Vcfile and Vcfolder](#)
- [Write Checkout of Vcfile and Vcfolder](#)

DSFA Selectables

- [Business Object Selectables](#)
- [Selectable Fields for Vcfile, Vcfolder and Vcmodule](#)
- [Selectable Fields for Abstract Vcfile and Vcmodule](#)

- [History for DesignSync File Access](#)

- [Find the ENOVIA Object ID from DesignSync](#)

- [Synchronize ENOVIA Updates with DesignSync](#)

- [Troubleshooting DSFA](#)

PowerView

- [About PowerView Pages](#)
- [Building and Linking a PowerView Page](#)
- [Parameters and Settings for Portal Objects](#)
- [Parameters and Settings for Channel Objects](#)
- [Parameters for Tab Command Objects](#)
- [Settings for Tab Command Objects](#)
- [URL Parameters Accepted by emxPortal.jsp](#)

JavaScript APIs

- [emxEditionableTable APIs](#)

- [Selecting Rows](#)
- [Expanding Rows](#)
- [Getting Cell Values by Row ID](#)
- [Getting Cell Values by Object ID](#)
- [Setting Cell Value by Row ID](#)
- [Setting Cell Values By Object / Relationship ID](#)
- [Enabling / Disabling Cell Edit by Row ID](#)
- [Enabling / Disabling Cell Edit by Object / Relationship ID](#)
- [Getting Current Cell Details](#)
- [Getting Parent ID](#)
- [Getting Child IDs](#)
- [Getting Child Column Values](#)
- [Getting Parent Column Values](#)
- [Reloading Cell Values](#)
- [Refreshing Table Columns](#)
- [Removing Selected Rows](#)
- [Complete Cell Information](#)
- [Displaying Validation Errors](#)

- [Markup APIs](#)

- [Load Mark Up](#)
- [Adding Rows to Selected Rows](#)

- [Miscellaneous APIs](#)

- [Load Custom Styles](#)
- [Disabling Check Boxes](#)

Techniques

- [Procedure for Creating an Attribute](#)
- [Procedure for Adding a Subtype](#)
- [Procedure for Adding an Attribute to a Type](#)
- [Procedure to Add a Trigger](#)

Best Practices

- [Business Objects](#)
- [Code Cleanup](#)
- [Database Access](#)
- [Documentation](#)
- [General](#)
- [HTML Tags](#)
- [i18n](#)
- [Izdate](#)
- [JavaScript](#)
- [JSP](#)
- [Memory Leaks \(IE\)](#)
- [Miscellaneous](#)
- [MQL Syntax](#)
- [Objects](#)
- [Standard Pages](#)
- [Statements](#)
- [StringBuffer](#)
- [Stylesheets](#)
- [URLs](#)
- [Variables](#)
- [WebSphere](#)

Dynamic UI Parameters

- [Administrative Object Parameters](#)

[Settings](#)
[URL Parameters](#)
[Parameters Automatically Passed to URLs](#)

Selectables

[About Selectables](#)
[Definition Objects](#)
[Relationship Direction](#)
[Selectable Fields for Applications](#)
[Selectable Fields for Attribute](#)
[Selectable Fields for Business Objects](#)
[Selectable Fields for Commands](#)
[Selectable Fields for Connections](#)
[Selectable Fields for Dataobjects](#)
[Selectable Fields for Dimensions](#)
[Selectable Fields for Expressions](#)
[Selectable Fields for Form Fields](#)
[Selectable Fields for Formats](#)
[Selectable Fields for Forms](#)
[Selectable Fields for Groups](#)
[Selectable Fields for Inquiry](#)
[Selectable Fields for Interface](#)
[Selectable Fields for Location](#)
[Selectable Fields for Menus](#)
[Selectable Fields for Person](#)
[Selectable Fields for Policy](#)
[Selectable Fields for Process](#)
[Selectable Fields for Product](#)
[Selectable Fields for Program](#)
[Selectable Fields for Query](#)
[Selectable Fields for Relationships](#)
[Selectable Fields for Role](#)
[Selectable Fields for Server](#)
[Selectable Fields for Store](#)
[Selectable Fields for Table](#)
[Selectable Fields for Table Columns](#)
[Selectable Fields for Type](#)
[Selectable Fields for User](#)
[Selectable Fields for Vault](#)
[Selectable Fields for Webreports](#)
[Selectable Fields for Wizard](#)
[Selectable Fields for Workflow](#)

Macros

[History and Common Usage](#)
[Available Macros](#)
[Macro Categories](#)
 [Intrinsic Macros](#)
 [Business Object Identification Macros](#)
 [Format Macros](#)
 [Lifecycle Check and Action Macros](#)
 [Wizard Variables](#)
 [Program Range Macros](#)
 [Dynamic UI Macros](#)
 [Connection Macros](#)
 [Workflow Macros](#)
 [JPO Macros](#)
[Trigger Macro Categories](#)
[Trigger Event Macros](#)
 [Attribute Event Macros](#)
 [Business Object Event Macros](#)
 [Connection Event Macros](#)
 [State Event Macros](#)
 [Process Event Macros](#)
 [Activity Event Macros](#)
 [Query Event Macros](#)
[Macro Processing and Command Syntax](#)

.NET Support for the Studio Customization Toolkit

[eMatrixClientXML.dll](#)
[.Net Sample Application](#)
 [Sample Code](#)
 [Building a Sample Application](#)
[.NET Limitations](#)

File Collaboration Server

[File Collaboration Server](#)
[FCS Architecture](#)
 [FCS Architecture/Configuration](#)
 [Other Supported Configurations](#)
 [Tickets and Receipts](#)

- [Added Security](#)
- [Stores, Sites, and Locations](#)
- [FCS Synchronization](#)
 - [Synchronizing Captured Stores](#)
 - [FCS Compressed Synchronization](#)
- [File Checkins](#)
 - [About File Checkins](#)
 - [One-Part Checkin](#)
 - [Two-Part Checkins](#)
 - [Checkin with Correlate](#)
- [About File Checkouts](#)
- [Replication](#)
- [Business Object Proxies \(BOPs\)](#)
- [FcsClient](#)
- [Large Files](#)
- [File Stream Filters](#)
 - [Java Class to Filter File Stream](#)
 - [Sample Filter](#)
 - [The Parameters File](#)
 - [Filter Calls](#)
- [Debugging](#)
 - [FCS Trace](#)
 - [FCSTools](#)
- [Custom JSP Pages for FCS](#)
 - [FCS with Custom JSPs](#)
 - [General Pages common.inc](#)
 - [Checkin Pages](#)
 - [Checkout Page](#)
 - [Error Page](#)
- [FCS Network Security](#)
 - [About FCS Network Security](#)
 - [About FCS Network Security Views](#)
 - [Implementing Out-of-the-Box Authentication Management](#)
 - [Implementing Custom Authentication Management](#)
 - [FCS Network Security Error Messages](#)

Coding Examples

- [Reading Property Values from the Cache](#)
- [Using Push and Pop to a Shadow Agent](#)
- [Setting a Target Page](#)
- [Checking that the User is Logged In](#)
- [Getting the User Context](#)
- [Looking Up an Object by Symbolic Name](#)
- [Opening an Object and Getting Information](#)
- [Reading Attributes](#)
- [Checking a User's Role](#)
- [Displaying MQL Notices](#)
- [Getting Administrative Object Names from Symbolic Names](#)
- [Getting Symbolic Names](#)

Index

Overview

This guide is a tool for programmers who are creating their own applications using Business Process Services, writing their own interface, or extending an existing application.

Important: Published examples in this document, including but not limited to scripts, programs, and related items, are intended to provide some assistance to customers by example. They are for demonstration purposes only. It does not imply an obligation for ENOVIA to provide examples for every published platform, or for every potential permutation of platforms and products and versions, and so on.

- [Before Reading this Guide](#)
- [Programming Skills Required](#)
- [Other Software](#)
- [Related Documentation](#)
- [Use of General ENOVIA Client Applications](#)

Before Reading this Guide

For information about system requirements, refer to the Program Directory for the version of the application you are installing.

Before you begin programming, review these guides:

- *Business Modeler Guide*. Review the Business Modeler concepts terminology, and processes.
- *Matrix Navigator Guide*. Review all information to become familiar with ENOVIA Live Collaboration.
- *ENOVIA Live Collaboration Installation Guide*. The ENOVIA Live Collaboration Server must be installed before installing Business Process Services. Information on installing the server is included in this guide.
- Javadocs for Business Process Services. Refer to the Javadocs as needed to learn the classes, methods, and properties in the Business Process Services packages. The Javadoc is located in .../server/doc/adkreference/index.html.

Note: Always refer to the current Program Directory for any changes since the publication of this manual.



Programming Skills Required

The information in this guide is intended for use by developers with a strong foundation in programming.

Specifically, the following skills are required:

- Knowledge of database management and Web servers
- Java, JavaServer Pages, and JavaScript programming
- Object-oriented programming techniques
- Programming in a multi-threaded environment



Other Software

To use Business Process Services, you must have several other software development tools. In some cases, two or more versions of similar products are available. You can select the one that best suits your needs.

Java Compiler

When you purchase the Java compiler, you also receive the Java Abstract Window Toolkit (AWT) and the Java API, which contain other Java classes needed for creating applications. The Java compiler (included with the Java Developer's Kit) from Sun version 1.3.3 or higher (available from <http://java.sun.com>) is required when working the ENOVIA Live Collaboration servlet Business Process Services.

Make sure you have the Java servlet class files (<http://java.sun.com/products/servlet/download.html>) and have the classes included in the CLASSPATH environment variable.

For detailed information on all the packages provided by Java, refer to your Java Compiler online help.

Some of the products useful for working on a PC platform that you may want to consider are:

- NetBeans
- Eclipse
- Visual Cafe from Symantec
- J++ from Microsoft
- JBuilder from Borland

Web Browsers

The browser is used for development and for viewing your applet or JSPs. You can use Mozilla, Firefox, or Microsoft Internet Explorer.

ENOVIA Live Collaboration Server

The ENOVIA Live Collaboration Server is the interface to the data stored in the Oracle database.

The ENOVIA Live Collaboration Server is described in the *ENOVIA Live Collaboration Installation Guide*. The ENOVIA Live Collaboration Server requires an application server when you use the Studio Customization Toolkit to create JSPs or servlets. For a list of the application servers that ENOVIA currently supports, see the *ENOVIA Live Collaboration Installation Guide*.

For client side applications and JPOs you only need the ENOVIA Live Collaboration Server; no application server is required.



Related Documentation

This section lists the documentation available for Business Process Services and ENOVIA products.

Also see "Installing Internationalized User Documentation" in the *Live Collaboration Administrator's Guide*.

Business Process Services Documentation

The Business Process Services installs with this documentation. The PDF versions of these documents are located in ENOVIA_INSTALL\Apps\BusinessProcessServices\ VERSION\pdf.

- *Live Collaboration Administrator's Guide*. This guide is for people in the host company who need to configure and customize ENOVIA products. It describes the schema that underlies the applications and how to configure it. It also describes all the properties available to configure Business Process Services.
- *Schema Reference Guide*. This guide is available in pdf format. The guide provides a reference for all of the schema that underlies ENOVIA products.
- *Application Exchange Framework User's Guide* and online help. This guide is available in pdf format and in html format as a context-sensitive online help system. It describes how to use features installed with the Application Exchange Framework portion of Business Process Services, such as history pages and pages accessed from the global toolbar. It also explains how to navigate through the user interface, such as how to use table pages. Users can access this help system by clicking the help button on any framework-specific application page or clicking AEF Help at the top of any application help page.
- *Common Components User's Guide* and online help. This guide is available in pdf format and in html format as a context-sensitive online help system. It describes how to use features installed with the common components portion of Business Process Services, such as the common document model and routes. Users can access this help system by clicking the help button on any common-specific application page or clicking Common Components Help at the top of any application help page.
- *Team Central User's Guide* and online help. This guide is available in pdf format and in html format as a context-sensitive online help system. It describes how to use features installed with the Team Central portion of Business Process Services, such as workspaces. Users can access this help system by clicking the help button on any team-specific application page.
- *Business Metrics User's Guide and online help*. This guide is available in pdf format and in html format as a context-sensitive online help system. It describes how to use features installed with Business Metrics portion of Business Process Services. Users can access this help system by clicking the help button on any metrics-specific application page.
- JavaDocs for Business Process Services. For descriptions of methods in framework packages and classes, see ENOVIA_INSTALL\Apps\BusinessProcessServices\ VERSION\Doc\javadoc.

ENOVIA Product Documentation

All ENOVIA products install with this documentation.

- User's Guide and online help for each application. Users access online help for an application by clicking the tool on the toolbar of every page. The user guide is in pdf format and requires Acrobat Reader to view. These guides are for the people who will log in and use any part of the application, including Administrators who use the profile management portions of an application to manage person and company profiles.
- Administrator's Guide for each application. Each application has a separate guide for company administrators who work with ENOVIA products. These are the same people who will use Business Process Services and the Studio Modeling Platform to configure an application. The administrator's guide for each application contains information unique to that application.

Additional Documentation

A Program Directory is available for each version of Business Process Services and the ENOVIA applications.

The program directory is a website that organizes all the release information for all Dassault Systèmes products for a given release. It contains information about prerequisites, installation, licensing, product enhancements, general issues, open issues, documentation addenda, and closed issues.

For instructions on installing Business Process Services and applications, see the *ENOVIA Live Collaboration Installation Guide* for the version of ENOVIA Live Collaboration that is required for your application.

Use of General ENOVIA Client Applications

Some of the instructions in this and other administrator's guides require the use of a general Matrix client navigator. These navigators should be used only by specially-trained administrators.

These are the general navigators:

- desktop version of Matrix Navigator (also known as the thick client)
- Web version of Matrix Navigator (also known as the thin client, PowerWeb, eMatrixApplet, and the Web Navigator)

It is important to restrict the use of these general navigator applications to only a few specially-trained business administrators and to only the purposes described in the *ENOVIA Live Collaboration Application Exchange Framework User's Guide* and product administrator's guides. ENOVIA applications run JavaBean code that requires data to have specific characteristics and conditions. For example, objects may have to have certain relationships defined, have specific values entered for attributes, be in specific lifecycle states, or be in particular vaults. When a person works within the ENOVIA product user interface, these data conditions are met. However, the general Matrix navigators are not necessarily aware of these conditions and therefore a person working within the general navigators can easily compromise data integrity.

Another reason to restrict access to the general clients is that certain actions have different results depending on where the action is taken. A command on a JSP page may include options (such as additional MQL clauses) to ensure that the operation is completed as the application expects, but a user in a general client has no guidance on what options should be chosen. For example, when a file is checked into ENOVIA Live Collaboration using a general client, the store set in the policy is used; when using an ENOVIA product to check in a file, the person or company default store is used regardless of the store set by the policy.

The general navigators must or can be used in situations such as:

- ENOVIA product features require data that cannot be created within the ENOVIA product user interface.
For example, some user profile information and template information must be created in a general navigator.
- Automated business rules and processes need to be configured, such as triggers and autonamers.
- Data needs to be investigated for troubleshooting, testing, or data conversion.

The general navigators should only be used in these situations, using the instructions provided in ENOVIA documentation, and only by specially-trained business administrators. Standard users of ENOVIA products should never be allowed to work with their data in a general navigator and external customers should never be given access to a general navigator. Also, using Studio Customization Toolkit applications or any programming interface that does not go through the applications bean layer has the potential to cause undesirable results within the ENOVIA product data.

Background Details for Developing Applications

ENOVIA Live Collaboration is an extremely open environment that allows you to standardize your business processes in a number of ways. The ENOVIA Studio Modeling Platform desktop applications allow you to define business processes with lifecycles, triggers, wizards, and workflows, and then implement them as program objects.

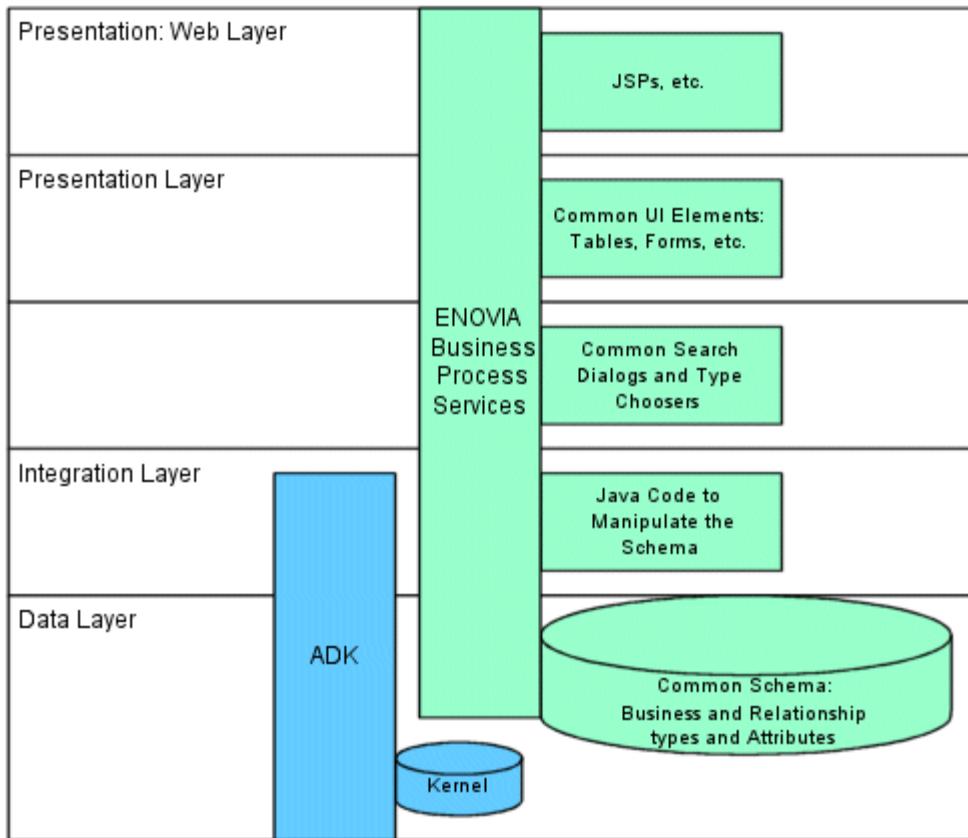
In this section:

- [About the Product Architecture](#)
- [Evolving Architecture](#)
- [About JSPs](#)
- [About Java Program Objects \(JPOs\)](#)
- [About JavaBeans](#)
- [Property Files, Language Strings, and Settings](#)
- [Studio Customization Toolkit Overview](#)
- [Setting up Java Packages](#)
- [Business Process Services Overview](#)

About the Product Architecture

ENOVIA Live Collaboration uses the Studio Customization Toolkit and Business Process Services to develop applications.

- [The Studio Customization Toolkit](#)
- [Business Process Services](#)



The Studio Customization Toolkit

The Studio Customization Toolkit is a low level Java API into the system. The Studio Customization Toolkit is responsible for manipulating the atomic object in the system such as Business Objects and Relationships and provides documentation, source code, sample files, and wrapper code that may be used as a reference for custom programming.

You can use the classes available in the applet or servlet jar files that are installed with ENOVIA Live Collaboration software (eMatrixAppletXML.jar and eMatrixServletRMI.jar). When you install the Studio Customization Toolkit, you get the javadoc for every ENOVIA Live Collaboration class that is included in all the .jar files, as well as source code that can be used as examples. For information about installing the Studio Customization Toolkit, see the *ENOVIA Live Collaboration Installation Guide*.

When you develop and compile against one of the servlet jar files, you can produce applications that use Java servlets, JPOs and JavaServer Pages (JSPs) to communicate with the database. The ENOVIA products (Engineering Central, and so on) are examples of applications that use the Studio Customization Toolkit. To build such an application, you first install the ENOVIA Live Collaboration Server you will run the programs on, and compile your code against the server's jar file. For example, if you are running the ENOVIA Live Collaboration Server, you need to compile against eMatrixServletRMI.jar.



Business Process Services

Business Process Services is a collection of the defined schema, as well as JPOs and Java Server Pages (JSPs) that can be used with the Studio Customization Toolkit to write your own applications.

Business Process Services is a combination of common UI elements, such as Tables, Forms, Common Search Dialogs, and Type Choosers. The common schema allows all the application to coexist and share data. The schema consists of instances of business objects, relationships, and attributes.

Business Process Services also includes the Java programming code needed to manipulate the pre-defined schema. Business Process Services makes Studio Customization Toolkit calls to work with the kernel.

The ENOVIA products (such as Engineering Central or Library Central) provide out-of-the-box process definitions on which to

base your implementations. They consist of JSP pages, JPOs and other application-specific schema. These applications provide flexibility in that they are highly configurable. First, they can be configured by making changes to the underlying schema, such as adding (or removing) a command to a menu or channel, adding (or removing) a column to a table, or adding (or removing) access to a column or command by a user, group, or role. Another way to configure the applications is via standard J2EE mechanisms. XML files define parameters for the applications which are then encompassed during J2EE deployment. If necessary, you can also modify existing JPOs or add new ones to extend the application as required.

Evolving Architecture

Business Process Services is an extendable object-oriented framework that leverages filters, inheritance, polymorphism, hooks, and a wealth of APIs.

- [Organization of Java Code](#)
- [Layers Design](#)

The framework is a web-based architecture based on J2EE, which hides the complexity of authentication, session management, transactions, and so on. Any changes must be backward compatible and offer a clear migration path. The Java Studio Customization Toolkit was the beginning of it all and has been around a long time. JSPs have been part of applications since their introduction, with Beans added to Business Process Services and JPOs added to the core in version 9.5.

Business Process Services continues to evolve as Java evolves and WILL accept such changes as the Sun Model 2 architecture, a controller framework (Struts, Springs, or JavaServer Faces), and the Front Controller pattern.

Because of this on-going evolution, different applications (and custom code in the field) are at different stages of adherence to the latest architecture.

Organization of Java Code

Java code in ENOVIA products falls into the areas described in this section.

- Java Server Pages (JSPs). About 200 web forms & tables and 100 web wizards
- Java classes (Beans). About 370 and growing.
- Java Program Objects (JPOs). About 150 pairs (base/derived) and growing.
- Studio Customization Toolkit. About 185 db classes, 54 utility classes and growing.

The Studio Customization Toolkit classes maintain a tight cohesion and loose coupling. Beans expose a higher abstraction than the Studio Customization Toolkit. Business logic Beans are the "business objects" (Business Delegate pattern) exposed to the JSP. These business objects never contain web-specific logic, so they can be freely used in stand-alone applications that do not run on an application server. Additional logic that does not belong to a specific type is placed into utility Beans.

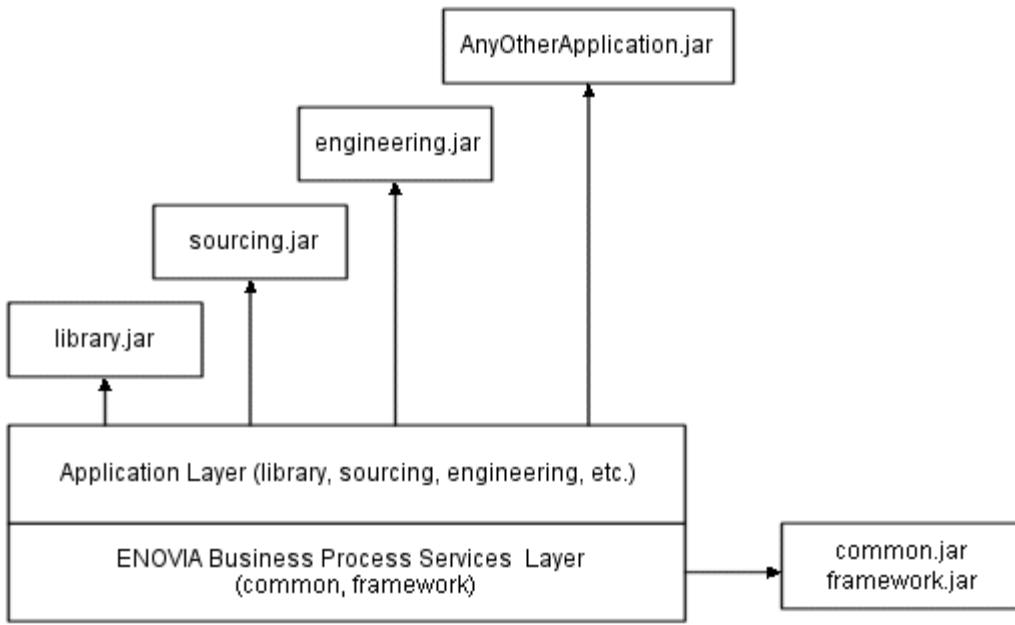
The grouping of logic is often determined by the nature of the functionality. For example, a Bean could represent an administration type, such as Policy. There might be another Bean used for referencing properties, either externally or internally. Another grouping of logic is found in the UI Beans which provide an API to the underlying UI components, such as tables, forms, and commands.



Layers Design

In a layers design pattern, each layer can only call downward. JSPs call Bean methods, which in turn call JPO methods, which in turn call Studio Customization Toolkit methods. Beans currently contain calls to Studio Customization Toolkit methods (since they were developed before JPOs existed) but most of these calls are isolated in the DomainObject base class.

The most used methods in DomainObject are findObjects(), getInfo(), and getRelatedObjects(). Beans represent Business logic (grouped by type), Utility logic (grouped by function), and access to UI components. Trigger logic should live in JPO methods. Trigger methods should be grouped into classes by type. Column support methods can be grouped into a companion JPO responsible for UI.



Each layer performs specific roles:

- **JSP (view)**. JSPs are packaged by application. They are for presentation only so they use tags and call Beans. This is known as "Business Delegate design pattern."
- **Bean (model front-end)** . Beans are no longer distributed in a single jar. They contain business logic, the (web) services API, and call JPOs. This is known as "Bridge design pattern."
- **JPO (model back-end)**. JPOs are installed by ENOVIA products. They implement business logic, call the Studio Customization Toolkit, and are used for triggers and UI component support.

The ENOVIA architecture uses a Model View Controller design pattern, where presentation is isolated from business logic. This design allows multiple views for a common model, and facilitates the management of code. ENOVIA highly recommends that your customizations adhere to this design pattern. The view portion is defined in the JSPs. The model portion is spread over Beans and JPOs. The ultimate goal is to have all business logic live in JPOs. ENOVIA has been moving business logic from JSPs into Beans, and now is moving it from Beans into JPOs.

Business logic is accessed through Beans. There is a type Bean for each business type, and Beans for some relationship types. The Part Bean provides services that one would expect to operate on Part objects. Currently, JSPs and stand-alone applications are consumers of these services. Eventually, a subset of these services will be turned into web services so that they can be accessed remotely, say from a portlet. The controller portion will be formalized through the addition of a Controller to be incorporated into the architecture.

About JSPs

JSPs are converted to servlets behind the scenes so coding of a JSP is simpler than coding for a servlet. JSPs combine HTML, JavaScript, and Java code.

See [Extending Applications](#) for more detailed information about JSPs. This introduction presents a baseline of how these methods, pages, and programs all relate to one another.

Whereas HTML is basically a document display format, JSPs provide the processing power that lets you create complex client/server applications for the Web and include dynamic substitutions within an HTML page.

JSP is an extension to the Java servlet technology from Sun, but is much different from ordinary Java in that JSPs are dynamically compiled and interpreted on the JSP-enabled application server. With Java servlets, you need a pre-compilation step, where you compile the text into bytecode which makes it more compact, faster to download and faster to interpret. Then the bytecode needs to be compiled and deployed. With JSP, the text of the Java code is interpreted and deployed immediately in one step. You can edit JSPs in pure text and instantly see the changes in the Web browser, just as you can with pure HTML.

In ENOVIA Live Collaboration, JSPs may be stored in the database as page objects or on disk. They are delivered through HTTP in response to the browser. JSPs directly call the business object interface of the Studio Customization Toolkit, using that interface to leverage RMI (Remote Method Invocation), which allows software components stored in the network to be run remotely.

JSPs can also include fragments from other page objects for reusable content. There are many reusable pieces in pure HTML that need to be referred to in multiple pages in an application: the navigation bar, parts of the action bar, the background image, the pieces of table definition that help organize the menu items, and so on. The fragments of HTML can be broken up into separate JSPs, and an include technique can be used to dynamically include the page objects that are needed. In this way, a change to an individual component, for example the navigation bar, can be made in a single resource and the whole application will respond because they are all using that shared fragment. This methodology provides the ability to dynamically model and change the look and feel of applications.

About Java Program Objects (JPOs)

A Java Program Object (JPO) is just another type of Program object that contains code written in the Java language. Generally, anywhere a Program object can be used, a JPO can be used.

The following topics are discussed:

- [JPO Code](#)
- [Separating Interface from Implementation](#)
- [Applications](#)

JPO Code

JPOs provide the ability to run Java programs natively inside the kernel without creating a separate process and with a true object-oriented integration feel as opposed to working in MQL.

JPOs allow you to write Java programs on top of the Studio Customization Toolkit programming interface and have those programs invoked dynamically.

When running inside the kernel, the programs share the same context and transaction as the thread from which they were launched. In fact, the Java programs are running inside the same Java Virtual Machine (JVM) as the kernel.

Java Program Objects are also tightly integrated with the existing scripting facilities.

Java code must be contained in a class. In general, a single class will make up the code of a JPO. Simply translating Tcl program objects into Java is not the goal. This would lead to many small classes, each containing a single method. The very object-oriented nature of Java lends itself to encapsulating multiple methods into a single class. The goal is to encapsulate common code into a single class.



Separating Interface from Implementation

The Java code in a JPO should be thought of as server-side (back-end) implementation code.

A JPO can be written to provide any logical set of utilities. For example, there might be a JPO that provides access to Administration objects that are not available in the Studio Customization Toolkit. In general, a JPO is associated with a particular Business Type. ENOVIA recommends that the name of the JPO contains the name of the Business Type (but this is not required). Access to a JPO should be done through an interface Bean.

You must manually create the JPO and write the methods inside it. Keep in mind that JPO code should be considered server-side Java; do not use Java GUI components in a JPO.



Applications

JPOs are ideally suited for use inside applications resulting in the advantage of moving Java code off the JSPs (away from the View and into the Model, considering the Model View Control design) and executing it on the appropriate application server (back-end) tier for optimum performance.

JavaBeans can be used on the Web server (middle) tier by JSPs as the interface to the JPOs.

The ENOVIA products ship with JPOs defined for most Business Types in the schema. Their style adheres to the guidelines found in this guide. These JPOs can be used as examples when writing your own JPO. Extensions to the ENOVIA products should use inheritance from these JPOs.

Each JPO is shipped as a combination of a base class and a derived class. All out-of-the-box (OOTB) business logic is placed in the base class. The derived class is available for extensions to the business logic. Since the system always refers to the derived class, custom extensions are picked up without needing to change OOTB logic. For example, a Business Type named Project would be represented by the 2 JPOs named emxProjectBase and emxProject (which extends emxProjectBase). Any custom business logic should be placed in the emxProject JPO. This form of application extension is less susceptible to problems associated with installing a new version of the application.

To further exploit the power of inheritance, the Business Process Services ships with 2 JPOs that serve as the base class for all type JPOs (business object and relationship) in the schema. The JPO emxDomainObject should be the base class for each JPO representing a business object type. The JPO emxDomainRelationship should be the base class for each JPO representing a relationship type.

About JavaBeans

JavaBeans encapsulate business logic that might otherwise appear on a JSP. The ultimate goal is to keep the Bean interface thin and push the business logic into the application server tier in the form of a JPO. The idea is to have the lightweight Bean (living on the Web tier) call the JPO to do all the work and then hold the resulting information (state) to serve up as needed to JSPs. JavaBeans are simply Java classes that support introspection and are easy to use within a JSP.

The `jsp:useBean` associates a JavaBean with the JSP and ensures that the object is available for the scope specified in the tag. The bound object can be referenced using the associated id from one or more JSPs (depending on the scope). The tag syntax is as follows:

```
<jsp:useBean id="name" scope="page|request|session|application"
beandetails />
```

Where beandetails is one of:

```
class="className"
class="className" type="typeName"
beanName="beanName" type="typeName"
type="typeName"
```

Using the "Hello World" example, if you assume there is a HelloWorld Bean to act as the interface to the JPO, then you might find the following usage in a JSP:

```
<jsp:useBean id="helloBean" scope="session" class="HelloWorld"
/>
<html>
<body>
<%
    helloBean.hello();
%>
</body>
</html>
```

This shows a HelloWorld Bean being defined and given an id of "helloBean". The `hello()` method simply calls the JPO to generate the "Hello World!" text.

```
public class HelloWorld implements Serializable
{
    public HelloWorld ()
    {
    }
    public void hello()
    {
        String[] init = new String[] {};
        String[] args = new String[] {};
        // establish a context ? <details not shown>
        int status = JPO.invoke(context, "Hello World", init,
"mxMain", args);
    }
}
```

Property Files, Language Strings, and Settings

ENOVIA products can be configured to fit your company's business processes using a combination of properties files, string resources and settings.

The following topics are discussed:

- [Properties Files](#)
- [String Properties](#)
- [Settings](#)

Properties Files

The emxSystem.properties file contains properties that apply to all applications, such as searching, file management, date management, and so on.

Each application has an additional properties file:

`emxAPPLICATION_NAME.properties`

where `APPLICATION_NAME` is the name of the specific application, such as EngineeringCentral or SupplierCentral. The application properties file contains default settings that affect that specific application. In Engineering Central, for example, you can change the properties that control how the Part Where Used page works (relationships used for searches, how many levels to expand, etc.).

Refer to the individual Administrator's Guides for information on setting properties.



String Properties

Each ENOVIA product has a string properties file that contains the text displayed in the user interface and email notifications. You can edit the file to change the text.

For example, you can edit an email notification to add a company-specific proprietary statement.

The naming convention for the base resource file (English) is:

`emxAPPLICATION_NAMEStringResource.properties`

For example, the string resource file for Engineering Central is `emxEngineeringCentralStringResource.properties`. The `emxFrameworkStringResource.properties` and `emxCommonStringResource.properties` string resource files provide text for pages available to all applications, such as the login, home, or lifecycle pages.

The naming convention for all internationalized versions of the StringResource files is:

`emxAPPLICATION_NAMEStringResource_LANGUAGE.properties`

where `LANGUAGE` is the standard abbreviation for the language, such as `fr` for French or `de` for German, in lowercase.

Refer to the Administrator Guides for information on editing string properties; refer to the *Live Collaboration Administrator's Guide* for information on localizing the string properties.



Settings

User interface elements, such as forms and tables, can be edited using the Business Modeler application or MQL. You can perform tasks such as renaming column headings, adding or removing columns from tables, or editing a form's layout.

Refer to the *Business Modeler Guide* or the *MQL Guide* for instructions.

Studio Customization Toolkit Overview

To build a client application, use the classes in eMatrixAppletXML.jar. Additionally, if you use the ENOVIA Live Collaboration Server without an application server and want to build client-only applications (an application with no servlets or JSPs), you can use eMatrixServletRMI.jar.

In the case of the ENOVIA Live Collaboration Server, this is sometimes referred to as "talking directly to RMI." You can compile this kind of code against either the eMatrixServletRMI.jar, or eMatrixAppletXML.jar. (A servlet-enabled Web or application server is required to run servlets and JSPs.)

C++ classes (eMatrixMQL) are also included when you install the Studio Customization Toolkit. The eMatrixMQL.h file provides the C++ programmer with the same functionality as the Java Client package provides to the Java programmer.

This section includes the following topics:

- [Signing Jar Files](#)
- [Overview of Building Java Applets](#)
- [ENOVIA Live Collaboration Java Class Packages](#)
- [Source Code Provided](#)

[Signing Jar Files](#)

When referencing FcsClient.jar or any other signed jar file, you must use the supplied certificate.

These certificates are in this directory:

\$ENOVIA_INSTALL/etc/certs

To sign a jar, *JARNAME*.jar for the FcsClient.jar (as an example), issue this command:

```
jarsigner -signedjar FcsClient.jar -keystore $ENOVIA_INSTALL/
etc/certs/releng.pk12 -storetype pkcs12 JARNAME.jar
```

To verify that the jar is correctly signed:

```
jarsigner -verify -keystore $ENOVIA_INSTALL/etc/certs
releng.pk12 -storetype pkcs12 -storepass releng -verbose
JARNAME.jar
```



[Overview of Building Java Applets](#)

The Studio Customization Toolkit includes sample code that you can use to build your own applications. You need to identify the Java class packages you want to use, link the specific classes into your own classes, and invoke the applet from an HTML page.

Throughout the section that describes the applet Studio Customization Toolkit, the terms *application* and *applet* are used. The following explains the difference between the two terms:

- Applications are not run within a Web browser. They do have access to the file system on the machine on which they are running. For example, you can use the Java Class packages to build Java applications, which are run just like any other desktop application.
- Applets are run within a Web browser. Applets cannot do anything to the local machine on which they are running (reading, writing or deleting files, for example) unless the applet code is signed with an unforgeable digital ID. Further, a user must give permission for an applet to perform any tasks outside the browser.

For a review of digitally signing a Java applet's files, see the home page (<http://www.suitable.com/docs/signing.html>) of Daniel T. Griscom (griscom@suitable.com).



[ENOVIA Live Collaboration Java Class Packages](#)

ENOVIA Live Collaboration .jar files include a set of Java class packages that you can use to build applications. The .jar files include other packages used at runtime but are not used to build and should not be called directly. These packages are not listed here and are not included in the Javadocs.

[eMatrixServlet and eMatrixApplet Jar File Packages](#)

This section lists the jar file packages in eMatrixServlet and eMatrixApplet.

- **db Package** . The db package consists of Java classes that implement and access ENOVIA Live Collaboration database objects. Database objects can be categorized as:
 - Administrative objects: attribute, form, format, group, page, person, policy, program, relationship, report, resource, role, and type objects.
 - Personal visual objects: query, set, cue, view, filter, toolset, and table objects.
 - Instance objects: business objects and relationships.

- **Resource Package** . The Resource package contains classes representing .gif images used in the ENOVIA Live Collaboration. The full source code for this Package is provided as part of the Studio Customization Toolkit.
- **VUI Package** . The VUI package provides Java binding for user interface components.
- **Util Package** . The Util package provides Java binding for utility functions.
- **ENOVIA Live Collaboration Servlet Package** . The Servlet package contains the ENOVIA Live Collaboration servlets.

eMatrixAppletXML Packages

This section lists the jar file packages in eMatrixAppletXML.

- **Client Package** . The Client Package provides Java binding of client functions that communicate with the ENOVIA Live Collaboration Server.
- **Common Package** . The Common Package consists of Java classes that implement common dialogs such as choosers. The full source code for this Package (including Class code and Method code) is provided as part of the Studio Customization Toolkit.
- **ENOVIA Live Collaboration Package** . The ENOVIA Live Collaboration Package consists of Java classes that implement application dialogs. The full source code for this package is provided as part of the Studio Customization Toolkit.



Source Code Provided

The Studio Customization Toolkit provides source code for the packages listed in this section.

- The Resource Package . Code is provided for .gif images that are included in various elements of the ENOVIA application user interface, such as images that make up buttons and icons and images that go on toolbar buttons.
- The VUI Package. Code is provided that implements Java binding for generic user interface components, including viewers, buttons, menus, icons, and toolbars.
- The Util Package. Code is provided for various utility functions, such as error message and problem reporting, searching, listing, and so on.
- The Common Package. Code is provided that implements common dialogs such as choosers and pattern windows: vault chooser, user chooser, type chooser, select pattern windows, and so on.
- ENOVIA Live Collaboration Package. . Code is provided that implements the Matrix Web Navigator. This applet can start the main application dialogs such as the flat browser, star browser, indented browser, state browser, basics, attributes, session context, find, formats, open set, and so on.
- Applet Package . Code samples are included for eight applets. Each one implements an eMatrixwindow within an HTML page, with the Business Process Services interface hidden from the user. For each applet, the classes, Java code, and an HTML page are provided. Applets are included to implement the following:
 - Attribute Dialog
 - Basics Dialog
 - FlatQuery
 - FlatSet
 - Formats Dialog
 - GetAttributes
 - Indented Browser
 - SimpleFind
 - Star Browser
 - State Browser
 - StatesPopup

Source code for Matrix Web Navigator is also included as a sample. Parts of its code are used throughout this guide as examples to demonstrate the key features of these packages and how you can use them to build applications.

- MQL Package. This applet displays an MQL screen in a Web page. Any valid MQL command may be executed, including those that update the database, such as create, approve or promote. When commands that involve external resources are executed, those resources exist on the ENOVIA Live Collaboration Server, not the client where invoked. For example, the MQL command:

```
print bus Assembly "MTC 12345" A select description current dump
"|" output 12345.txt
```

will create the text file on the ENOVIA Live Collaboration Server host machine in the ENOVIA Live Collaboration Server's directory, not on the machine running the applet. Similarly, external program files must reside on this host in this directory if they are to be executed from the MQL applet.

Setting up Java Packages

In general, the Java class packages should be set up and compiled as described in this section.

This section includes the following topics:

- [Java Package Structure](#)
- [Compiling Java Classes](#)

Java Package Structure

This section defines the directory structure of the Java packages.

`company/classes/product/package/*.class`

where:

Path Element	Definition
company	The company name.
/classes	Indicates that the directories below this contain class files.
/product	The application product name.
/package	These subdirectories are named for the class packages and they contain the class files.
/*.class	The various class files.

The ENOVIA Live Collaboration classes are provided in this structure:

```
matrix/classes/matrix/client/*.class  
matrix/classes/matrix/resource/*.class  
matrix/classes/matrix/db/*.class  
matrix/classes/matrix/vui/*.class  
matrix/classes/matrix/util/*.class  
matrix/classes/matrix/common/*.class  
matrix/classes/matrix/matrix/*.class
```

You should customize the structure to indicate your company and product, and any packages that you create. For example:

```
acme/classes/widget/client/*.class  
acme/classes/widget/db/*.class  
acme/classes/widget/vui/*.class  
acme/classes/widget/util/*.class  
acme/classes/widget/common/*.class  
acme/classes/widget/matrix/*.class  
acme/classes/widget/acmepackage1/*.class
```

You should create a similar structure for your Java source code. Replace the `classes` subdirectory with `source` so that the directory structure for your Java source code will be familiar. The source Java files contain the methods in the classes used for the implementation of the classes. For example:

```
acme/source/widget/client/*.java  
acme/source/widget/db/*.java  
acme/source/widget/vui/*.java  
acme/source/widget/util/*.java  
acme/source/widget/common/*.java  
acme/source/widget/matrix/*.java  
acme/source/widget/acmepackage1/*.java
```



Compiling Java Classes

When you compile using your Java class packages, you must specify the exact order in which the compiler searches for class files.

Specifying the search order is typically done through the environment variable `CLASSPATH`. Alternatively, if you work in a development environment like Symantec Cafe, `CLASSPATH` may be modified in the project file settings.

ENOVIA Live Collaboration classes are contained in the file `ENOVIA_INSTALL/java/classes/eMatrixAppletDownloadXML.jar`, which can be used to compile against. For example, use the following `CLASSPATH` statement:

```
CLASSPATH = ENOVIA_INSTALL/java/classes/eMatrixAppletXML.jar
```

If multiple .jar files exist, use `CLASSPATH` to specify the search order. For example, if there are two files called `widget.class` (one in the `acme.jar` file and one in the `eMatrixAppletDownloadXML.jar` file), and the one in `acme.jar` is the one you want to use, you must indicate this in the `CLASSPATH` statement. To search the `acme.jar` file first, specify the `acme` path before the

`matrix` path in the `CLASSPATH` statement:

```
CLASSPATH = /acme/classes/acme.jar;ENOVIA_INSTALL/java/classes/  
eMatrixAppletXML.jar;...
```

The compiler will search the `acme` path first and locate the `widget.class` file in the `acme.jar` file. The `widget.class` in `eMatrixAppletDownloadXML.zip` will not be used.

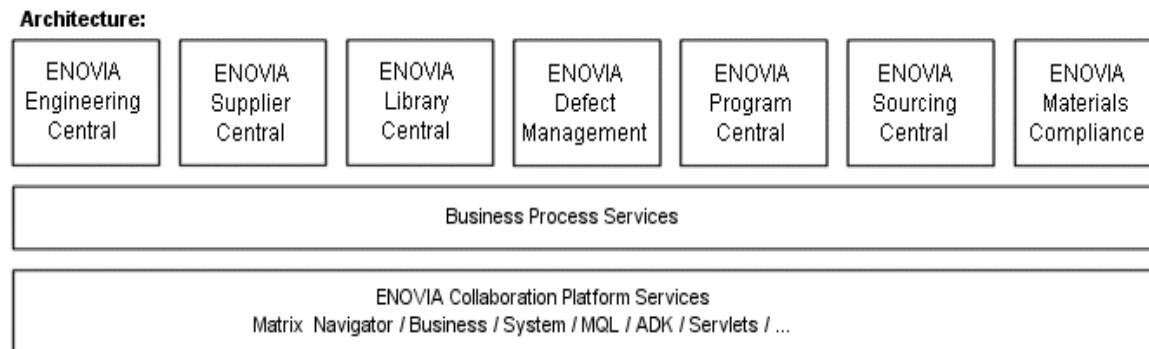
Business Process Services Overview

Business Process Services contains schema for all the ENOVIA products and is the framework, or foundation, for those products. It includes the programs and JavaServer Pages needed to construct the user interface shared by all the applications. The framework must be installed before you can install any ENOVIA product. The framework can also be used as the basis for creating your own applications.

This section includes the following topics:

- [Business Process Services Components](#)
- [ENOVIA Product Components](#)
- [Dynamic UI Components](#)
- [Database Objects](#)
- [Application Jar Files](#)
- [Naming Conventions](#)

This diagram describes the framework architecture:



Java code is in:

- framework.jar
- common.jar
- engineering.jar
- other application-specific jars...

The above graphic shows a sampling of ENOVIA products. All ENOVIA applications rely on the same architecture.

Business Process Services Components

This section lists the components Business Process Services provides to support the ENOVIA applications.

The Business Process Services contains these schema items as defined in the *Schema Reference Guide*:

- associations
- attributes
- business objects
- commands
- formats
- groups
- menus
- persons
- policies
- programs, including Java Program Objects (but not those specific to an application)
- relationships
- roles
- stores
- types
- vaults
- wizards (for administrative uses only, not for business processes)



ENOVIA Product Components

Each ENOVIA product contains the items listed in this table.

Application Components

Item	For information, refer to:
Web pages used by the application's users	The user guide that accompanies the application.
Programs specific to the application	To configure programs and for descriptions of utility trigger programs, see "Configuring Using Schema" and "Triggers" in the <i>Live Collaboration Administrator's Guide</i> . For application-specific trigger programs, see the Administrator's Guide that accompanies the application. The Administrator's Guides are located in ENOVIA_INSTALL\Apps\APP_NAME\VERSION\pdf For information on how to call the included JavaBeans in your custom applications, see the Javadocs located at: ENOVIA_INSTALL\Apps\APP_NAME\VERSION\Doc\javadoc
Other administrative objects specific to the application, such as formats	The Administrator's Guide for the application.
Business objects that accomplish system-related tasks, such as objects for automatically-naming objects and for executing trigger programs	For general information on how the objects function and how to configure them, see "Configuring Business Process Services Functions" and "Configuring Using Schema" in the <i>Live Collaboration Administrator's Guide</i> . For a list of the objects included in the application, see the Administrator's Guide that accompanies the application.

Some applications, such as Materials Compliance Central, install other components that may be shared between applications. When this is the case there are 2 directories installed under ENOVIA_INSTALL\Apps, one which includes "base" in the name, such as MaterialsComplianceBase. These base directories are where the doc directories can be found.

For example: c:\enovia\studio\Apps\APP_NAME\VERSION\pdf.

The documentation that describes the common components installed with Business Process Services can be found in the ENOVIA_INSTALL\doc directory.



Dynamic UI Components

Whether building a new application from scratch or extending an existing application, the UI components significantly reduce coding effort and make your code easier to upgrade.

The application interface consists of the following:

- Application Login
- Navigator Page and Toolbars
- Global Toolbar
- Menus
- Global/Generic Search component
- Application-specific Search Pages
- View Table Pages
- Mass Editable Configurable Tables
- Structure Browser Tables
- Navigation Trees / Structure Navigator Pages-.
- Configurable Web Forms
- Configurable Type Chooser Forms
- Configurable Vault Chooser Forms
- Calendar Popup Forms
- PowerView Configurable Pages (Portal Pages)
- Internet Portals



Database Objects

Business Process Services uses several database objects to support applications.

Business Process Services uses these database objects:

- Pre-defined schema (administrative and business objects)
- Program (Tcl and JPO) objects
- User Interface objects

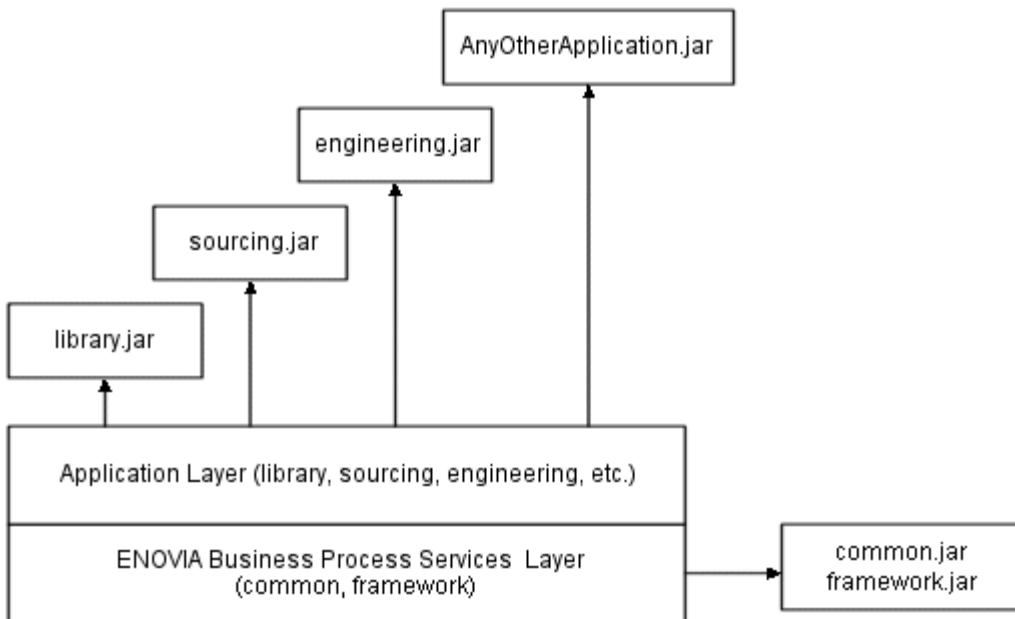
The schema files contain these items:

- JPOs contain business logic.
- XML files contain schema definition and rules.
 - The XML definition file contains the actual schema definition for types, attributes, relationships, and so on.
 - The XML manifest file lists what schema is required by an application or functional area.
 - The mapping file drives the installation wizard and defines prerequisites.
 - The conversion file describes what version a conversion should run at and specifies the prerequisite schema.

Application Jar Files

Each application installs its own jar file that contains Beans and utility classes and the Business Process Services installs multiple jar files. This installation scheme allows ENOVIA products to be updated without having to update everything.

You can look at the version field in the manifest files for each jar to figure out what versions you currently have installed. A pictorial representation of the multiple-tier jar hierarchy is shown below.



Business Process Services is responsible for these jar files: framework.jar and common.jar.

The framework jar file contains the base classes DomainObject and DomainRelationship, as well as several interfaces and classes needed to support these base classes. This jar file also contains a number of utility classes.

The framework jar file also contains custom tags that make up our custom tag library, and classes that wrap the dynamic UI components (often referred to as the UI Beans).

To promote reuse of business logic, a common jar file has been established independent of any application. The common jar file contains domain Beans that are shared by more than one application. The common jar file also contains utility classes shared by more than one application.

Jar Name	Packages
framework	com.matrixone.apps.domain, com.matrixone.apps.domain.util com.matrixone.apps.framework.ui, com.matrixone.apps.framework.taglib
common	com.matrixone.apps.common, com.matrixone.apps.common.util

There are strict rules on the dependencies allowed among the framework, common, and application tiers (as shown in the figure). Framework JSPs, JPOs, and Beans can only reference the framework jar files. The common components JSPs, JPOs, and Beans can only reference the common components and framework jar files. Each ENOVIA product's JSPs, JPOs, and Beans can only reference their own jar file and the common components and framework jar files.

An ENOVIA product can never reference another ENOVIA product's jar file.

Naming Conventions

The ENOVIA products allow you to use your exact business terminology. Names are case-sensitive and spaces are allowed. You can use complete names up to a maximum of 128 characters. Leading and trailing spaces are ignored.

The naming conventions for the Business Process Services ensure that:

- Programs refer to each other correctly.
- Business process application programs can be distinguished from non-business process application programs.

- It is easy to identify the purpose of the file and the application it belongs to.

Some object names installed with the Business Process Services are prefixed with "eService" and the version number of the installation. The installation program adds this prefix to prevent name collisions between objects in the framework and objects in the existing database. For example, if you install version V6R2011x BPS onto a database that contains a Part type, the installation program renames the Part type within the framework to "eServiceV6R2011x~Part". Other administrative objects within the framework that refer to the type--such as attributes for the type, relationships that connect the type, and policies that govern the type--refer to the framework Part type, now named "eServiceV6R2011x~Part." The installation makes no change to the existing Part type.

To list the name collisions the system found during installation, open the file named `installFrameworkVERSION.log`, where `VERSION` is the software version number, located in `ENOVIA_INSTALL\Apps\BusinessProcessServices\VERSION`. The file also lists all the administrative objects installed or modified during installation. For instructions on configuring objects in the framework, using existing objects instead of the objects installed with the framework, and for information about the installation process, see the *Live Collaboration Administrator's Guide*.

UI Basics

The dynamic user interface enables you to modify administration objects instead of changing code in JavaServer Pages or JavaBeans.

In this section:

- [Dynamic User Interface](#)
- [User Access to UI Components](#)
- [Configuring Styles](#)
- [Main Navigator Page](#)
- [Adding a Custom Application](#)

Dynamic User Interface

You can define strings used by the dynamic UI components using macros and select expressions.

In this section:

- [About the Dynamic User Interface](#)
- [Naming Conventions for UI Administrative Objects](#)
- [Using Macros and Expressions in Dynamic UI Components](#)
- [Internationalizing Dynamic UI Components](#)

About the Dynamic User Interface

The dynamic user interface is used to configure or add custom features to existing applications, or develop a new application.

The dynamic user interface lets you:

- Configure existing applications that are built using the dynamic user interface.
- Add menus and features to existing applications.
- Create a custom application that contains the interface.

The dynamic user interface enables you to accomplish these functions by modifying administration objects instead of changing code in JavaServer Pages or JavaBeans.

Naming Conventions for UI Administrative Objects

This section lists naming conventions used for UI objects installed with the framework.

The naming convention for most objects includes a three-letter abbreviation for the application that uses the object (for example, ENC for Engineering Central). These abbreviations are listed in the following table.

Application	3-Letter Abbreviation
Business Process Services	AEF, APP, or TMC
Defect Management	DFT
Engineering Central	ENC
Library Central	LIB
Materials Compliance Central	MCC
Product Line	PRC
Program Central	PMC
ENOVIA Requirements Central	RMT
Sourcing Central	SCS
Supplier Central	SUP

If you create custom objects, create your own abbreviation based on your custom application name. These naming conventions apply to everyone creating custom objects, including customers, partners, and ENOVIA Professional Services. The only objects that do not use the three-letter abbreviation are the top-level menu objects and the menu objects for the root node of trees. Menus for tree nodes use the symbolic name of the object type (type_Part). Do not include spaces in the names for UI administrative objects.

Admin Object Type, Usage	Convention	Examples
Menu objects for menu types (installed with framework)	Menu type name	My Desk (displays as ) Actions Toolbar
Menu objects for application submenus within a menu type (installed with applications)	Three-letter standard abbreviation for application name, followed by the menu type with no spaces.	ENCMyDesk TMCActions
Menu objects for root node of trees	Symbolic name of the tree type or the type's parent type.	type_Part type_DrawingPrint
Menu objects for toolbars	Three-letter standard abbreviation for application name, followed by the name of the action or feature the toolbar is used for, followed by "ToolBar", with no spaces.	ECBOMListToolBar TMCProjectsToolBar APPRouteSummaryToolBar SPCTechSpecActionsToolBar
ENOVIA Portal objects for object PowerView	Three-letter standard abbreviation for application name, Feature name, "PowerView".	SCSRFPowerView
Channel objects	Three-letter standard abbreviation for application name, followed by feature name, followed by "Channel".	SCSRFQLineItemsChannel
Command objects for channel tabs	Three-letter standard abbreviation for application name, followed by feature name.	SCSRFQLineItemsAttachments
Command objects for toolbar items and tree categories on pages	Three-letter standard abbreviation for application name, followed by the action or feature name, followed by the type of command (Actions, MyDesk, Tree, Toolbar, or TreeCategory).	ENCCreatePartActions TMCEditProfileToolbar AEFLogoutToolbar
Table objects	Three-letter standard abbreviation for application name, followed by the feature name.	ENCBOMList TMCWorkspaces SCSBuyerDesks
Inquiry objects	Three-letter standard abbreviation for application name, followed by the list name or the feature the inquiry is for.	ENCCBOM TMCPersons

		SCSPackages
Web Form objects	Three-letter standard abbreviation for application name, followed by the object type or feature name.	ENCPart SCSBuyerDesk
Web Form fields	Name of the field	Name Description Part Classification Originated By

Do not use abbreviations already used by ENOVIA products when creating custom objects:

Using Macros and Expressions in Dynamic UI Components

Many strings used in the definition of dynamic UI components (such as label values, hrefs, and settings) can contain embedded macros and select clauses.

- [About Macros and Expressions](#)
- [Directory Macros](#)
- [Select Expression Macros](#)
- [Other Macros](#)

About Macros and Expressions

The \$<> delimiters identify macro names. The system evaluates macros at run-time and before displaying in the UI.

Strings defined using macros and expressions cannot be internationalized.

Strings can include select clauses that are evaluated against the appropriate business object at run-time. The \$<> delimiters identify select clauses. Select clauses generally use symbolic names and are preprocessed to perform any substitutions before submitting for evaluation. This example shows a macro in the href definition, a macro in the Image setting, as well as a select clause in the label definition of a tree menu (associated with a Line Item object):

```
MQL<2>print menu type_LineItem;
menu type_LineItem
description
label '$<attribute[attribute_EnteredName].value>'
href '${SUITE_DIR}/LineItemDetailsFS.jsp'
setting Image value ${COMMON_DIR}/iconSmallIRFQLineItem.gif
setting Registered Suite value Sourcing
children
command SCSAttributeGroup
command SCSAttachment
command SCSSupplierExclusion
command SCSUDA
command SCSSLineItemHistory
```

The following example shows a typical business object macro being used in the label definition of a tree menu associated with a Company object (the tree menu shows as the Categories menu in the UI):

```
MQL<3>print menu type_Company;
menu type_Company
description
label '$<name>'
href '${COMMON_DIR}/emxForm.jsp?
form=type_Company&toolbar=APPCompanyDetailsToolBar&HelpMarker=emxhelpcompanyproperties&formHeader=emxComponents.'
setting Image value ${COMMON_DIR}/iconSmallOrganization.gif
setting Registered Suite value Components
children
command APPBusinessSkills
command BusinessUnit
command PMCCalendar
command APPCapabilities
command APPCollaborationPartners
command Department
command APPFormats
command Location
command APPPlant
command People
command Regions
command APPSubsidiary
command APPCompanyImageManager
command APPCurrencyExchangeRates
```

Surround macros with quotes to ensure proper substitution for values that contain spaces.



Directory Macros

The following table lists the directory-specific macros that you can use in parameters and settings for dynamic UI components.

If you do not specify a path for a file, the system looks in the registered directory (as specified in emxSystem.properties for the registered application). If there is no registered directory, the system assumes the file is in the current directory.

Macro Name	Use to have the system look for the file in:
\${COMMON_DIR}	The "common" directory obtained from the emxSystem.properties file. The key used is eServiceSuiteFramework.CommonDirectory = common.
\${ROOT_DIR}	The ematrix root directory obtained from emxSystem.properties using the key eServiceSuiteFramework.RootDirectory = .
\${SUITE_DIR}	The application-specific directory obtained from emxSystem.properties based on the Registered Suite setting. For example, the key used for "Engineering Central" is eServiceSuiteEngineeringCentral.Directory = engineeringcentral. Suite directories use the original application names.
\${COMPONENT_DIR}	The directory that contains common components, which is ematrix/component by default. The directory is obtained from eServiceSuiteFramework.ComponentDirectory = component in the emxSystem.properties file.



Select Expression Macros

You can define Select expression macros as \$<SELECT EXPRESSION>, where the select expression can be any valid MQL select statement.

You can use Select expression macros in labels for configurable forms and in expression parameters.

Macros can be used as labels in headers for forms, but cannot be used in headers for tables or structure browsers.

Select expressions are evaluated at runtime against the current business object ID and relationship ID that is passed in. Some examples include:

- \$<type>
- \$<name>
- \$<revision>
- \$<attribute[\$].value>
- \$<attribute[FindNumber].value>
- \$<from[relationship_EBOM].to.name>



Other Macros

You can also use the table selected count macro.

Macro Name	Description
\${TABLE_SELECTED_COUNT}	Substitutes the number of rows selected in the current table, including those selected on other pages within a paginated table. Use in a confirmation message for a toolbar item to let user know how many rows are selected. Define the confirm message using the Confirm Message setting for a toolbar item command.

Internationalizing Dynamic UI Components

The design of the dynamic UI components supports the ability to internationalize or localize text displayed on UI components.

If you enter standard text as the value or a macro, the text cannot be internationalized. When displaying the UI, the system looks in the string resources file(defined in emxSystem.properties) for the specified string property.

You can internationalize the following text within UI components:

- Page header labels
- Table headers and repeat headers
- Alt text on icons and images
- Table page filter list option
- Types selected in the configurable Type Chooser
- Labels for form page fields
- Form field values

For information on internationalizing other text within the applications, such as administrative type values and online help, see "Internationalizing the ENOVIA Products" in the *Live Collaboration Administrator's Guide*.

1. If creating a new application, create a new string resources file.

The format for the string resource file name is:

`eServiceSuiteAPPNAMEStringResource.properties`

where `APPNAME` is your application name.

2. In this file, define a property for each string using this format:

`emxAPPNAME.STRING.ID = TEXT TO DISPLAY`

where:

- `APPNAME` is your application name
- `STRING.ID` is the identifying name used in the dynamic UI component to identify this string
- `TEXT TO DISPLAY` is the actual text

3. If developing a new application, add the mapping details for your string resources file to the emxSystem.properties file.

Example property setting in emxSystem.properties for Engineering Central:

`eServiceSuiteEngineeringCentral.StringResourceId =
emxEngineeringCentralStringResource`

For Library Central:

`eServiceSuiteLibraryCentral.StringResourceId =
emxLibraryCentralStringResource`

4. When defining a UI component, enter a string resource ID property name as the value for the text or label and add that string resource to the stringresources.property file.

User Access to UI Components

The dynamic user interface defines access more finely than the application level so you do not assign access to an application.

In this section:

- [Access Checks for Individual Components](#)
- [Search Page Access](#)
- [Access Checks for Messages](#)
- [Access Parameters and Settings](#)
- [Sample JPO for Controlling Access](#)

Access Checks for Individual Components

The system provides four levels of access control for toolbar items, menu commands (commands in the My Desk menu, Actions menu, and global toolbar), tree categories, table columns, and form rows.

You define access to menu items, toolbar items, tree categories, table columns, and form rows (fields) using the administrative object that defines each of these components. The emxNavigator.jsp displays the submenu for an application if the user has access to at least one command object in the submenu. If the user does not have access to any command objects in the submenu, the submenu heading is not displayed.

To set these access checks, add the setting or parameter for the appropriate administrative object. For example, to control access for a toolbar item, edit the command object that represents the link. For details, see [Access Parameters and Settings](#).

Assuming all possible checks are enabled, the sequence of access checks are as follows:

1. Role-based. Users assigned to the Access tab for the relevant administrative object. These users are usually roles but can be persons and groups. When a role or group is assigned access, access extends to all child roles and groups. This access is the default or minimum access control.
2. Access Mask setting on the administrative object. Specifies the accesses the user must have for the current business object in order for the component to be displayed.
3. Access Expression setting on the administrative object. An expression whose value is a valid context-based expression that is evaluated at runtime.
4. Access Program and Access Function settings on the administrative object. The name of a JPO program and method that checks access.

The system performs the access checks in an all or nothing manner. That is, if all the access checks pass, access is granted. If at least one of them fails, the UI component is hidden.

Search Page Access

Clicking Search from the global toolbar opens the page defined at the top of the Search menu, AEFGlobalSearch. When the framework is installed, the default shows the AEFFullTextSearch search page.

1. Once all applications have been installed, rearrange the Search menu so the most often used Search page is at the top of the menu. ENOVIA suggests that you move the Saved Searches command to the last item in the Search menu, since Saved Searches have no items listed until each user creates and then saves their own searches.

For details, see [Global Search Menu](#).

2. Assign the URL parameter `defaultSearch` to the search command you want as the system default.

When set, all users with access see that search page when they click Search from the global toolbar. If any users do not have access, they get the first search in the list they have access to. For details, see the *Live Collaboration Administrator's Guide*

Access Checks for Messages

A discussion consists of an initial message and the replies to that message. Access is initially determined based on whether or not the logged in user has access to the discussion's parent object.

Access is also determined by whether or not the logged in user is:

- Internal. Employees of the host company (and any business units, divisions, or departments)
- External. Employees of other companies (such as suppliers or customers) who have been granted access to the ENOVIA product

Internal users can access all public and private messages (if they have access to the parent object); external users can only access public messages (if they have access to the parent object).

Note: External users assigned to the Private Discussion association can only access those messages where the parent object of the discussion is connected to that user's organization using the Design Responsibility relationship.

The initial message can be created as public or private. If public, an internal user can create either a public or private reply to the message. If private, external users have no access to the discussion. If an internal user creates a private reply to a public message, or future replies to that reply will be private and cannot be changed to public. If an internal user wants to create a public reply, that user must select a prior public message or create a new discussion.

Public messages use the Message policy; private messages use the Private Message policy. Refer to the *Schema Reference Guide* for details of these policies.

Access Parameters and Settings

This table describes the parameter and settings that control access to menu commands, toolbar items, tree categories, table columns, and form rows.

Parameter/Setting	Description	Accepted Values/Examples
Access tab (Business Modeler tab / user MQL command)	<p>Use to specify the persons, roles, and groups who can access the component. When you assign a role or group, all child roles/groups also receive access. To make the component available to all users, regardless of role/group assignments, choose All.</p> <p>If no users are assigned access, the system assumes all users have access.</p>	Names of group, role, person administrative objects. Or All (default)
Access Expression setting	<p>Controls access to the component based on a valid expression. The system evaluates the expression at runtime. If the expression evaluates to True and no other access control prevents access, the component is shown. Depending on the type of expression defined, the program may or may not need a valid objectID.</p> <p>Make sure that an objectID is available before configuring this setting for configurable toolbar menus and commands.</p> <p>When you use the setting for commands connected to the AEEGlobalToolbar (My Desk, Actions, Tools), the system evaluates the expression on the person business object. It does not rely on any objectId to be passed in as a URL parameter.</p> <p>In cases where no objectId is passed, such as a table page originating from My Desk, the Access Mask setting is ignored. Similarly, the IconMail tree, which does not have an objectId, does not use this setting if defined.</p>	<ul style="list-style-type: none"> \$<attribute[attribute_Weight].value>? > 100 The toolbar item displays only if the Weight attribute on the business object is greater than 100. \$<attribute[attribute_Originator].value>== owner <p>For menu command objects:</p> <ul style="list-style-type: none"> context.user.name=="Test Everything" context.user.isassigned[Employee] == TRUE
Access Function setting	<p>The name of the JPO method to invoke in the JPO specified for the Access Program setting. The Access function gets the input parameter as a HashMap that contains all the request parameters passed into the JSP page. The JPO method must return a Boolean object. If the returned value is true and no other access control prevents access, the component displays. If false, it is hidden.</p> <p>To see a sample JPO method to control access, see Sample JPO for Controlling Access.</p> <p>Use this setting to evaluate access independent of users' roles. For example, suppose a link should only be shown to users who are employees of the host company. The JPO and method might check the user's company and display the link only if the user is from the host company.</p>	The name of an access check method in the JPO specified in the Access Program setting, such as: emxAccessCheck()
Access Mask setting	<p>Specifies the accesses the user must have for the current business object for the component to display.</p> <p>When you use this setting for commands connected to the AEEGlobalToolbar (My Desk, Actions, Tools), the expression is evaluated on the Person business object. It does not depend on any objectId to be passed in as a URL parameter.</p> <p>When you use the setting for commands in page toolbars, menus, trees, table columns, and form fields, the system evaluates the access mask on a specific business object. This business object is available only when the objectId is passed in as a URL parameter to the JSP.</p> <p>Make sure that an objectId is available before configuring this setting for configurable toolbar menus</p>	Any set of accesses, separated by a comma. For example: Modify Delete ToConnect ToDisconnect FromConnect FromDisconnect Modify,Delete FromConnect,FromDisconnect

	<p>and commands.</p> <p>In cases where no objectId is passed, such as a table page originating from My Desk, the Access Mask, the system ignores this setting. Similarly, the IconMail tree, which does not have an objectId, does not use this setting if defined.</p> <p>If the user does not have all the specified accesses in the business object's policy for the current state, the system hides the component. If the user has the access and no other access control prevents access, the system displays the component.</p> <p>You can specify multiple accesses by separating the accesses with a comma.</p>	
Access Program setting	<p>Controls access to the UI component based on the output from a method in the specified JPO program. You must define the program in Business Modeler. This setting requires that you also specify the Access Function setting. If Access Function is not set, the system ignores the Access Program setting.</p> <p>The following input values are required for the program:</p> <ul style="list-style-type: none"> • A list of all of the request parameters in a HashMap • Method name as a string • JPO Program Name as a string • Context <p>The output must be a Boolean.</p>	<p>Name of a JPO defined as a program object, such as:</p> <p>emxAEFCollectionAccess</p>

Sample JPO for Controlling Access

You can define a JPO method for controlling access to UI components.

The following settings are required for the component whose access you want to control:

- Access Program=<JPO Name>
- Access Function=<JPO method name>

Access Function gets the input parameter as a HashMap, which contains all the request parameters that were passed into the Form page. The method can use this Map to obtain any request parameters that are required for processing and deciding access.

This method returns a Boolean. The value is true if the component is accessible to the current user context and the given object, if any. Otherwise the value is false and the component is hidden. Here is a template for a JPO function that controls access for a form field:

```
public static Boolean methodName(Context context, String[] args)
throws Exception
{
    // Define a boolean to return
    Boolean bFieldAccess = new Boolean(true);
    HashMap requestMap = (HashMap) JPO.unpackArgs(args);
    // Get the parameter values from "requestMap" - if required
    String objectId = (String) requestMap.get("objectId ");
    String relId = (String) requestMap.get("relId ");
    String languageStr = (String) requestMap.get("languageStr");
    // Do the necessary processing to define access and assign
    true/false to bFieldAccess
    ....
    bFieldAccess = new Boolean(true);
    // (or)
    // bFieldAccess = new Boolean(false);
    return bFieldAccess;
}
```

Configuring Styles

You can edit the provided style sheets to customize the appearance of an application

All cascading style sheets and images are in ematrix/common.

Changing a style sheet file will affect ALL pages that use it. Make sure that any changes you make will work on all pages that reference it. To change the look of an individual page, you must create a special style sheet for that purpose.

The graphics that display in the banner are defined in emxNavigator.jsp. If you need to change the images, you must customize this JSP using a CustomFilter as described in [Extending Applications](#).

- [emxUIChooser.css](#)
- [emxUIDialog.css](#)
- [emxUIForm.css](#)
- [emxUILifeCycle.css](#)
- [emxUIList.css](#)
- [emxUIListPF.css](#)
- [emxUINavigator.css](#)
- [emxUIProperties.css](#)
- [emxUISearch.css](#)
- [emxUIToolbar.css](#)
- [emxUITree.css](#)
- [emxUIWizard.css](#)
- [emxUIWorkflow.css](#)

emxUIChooser.css

To change how choosers appear, edit the style sheet file emxUIChooser.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
body { }	Default Background Appearance	Background

Default background appearance

The screenshot shows a search dialog titled "Search: People". The dialog has a light gray header bar with "Search: People" and standard window controls. Below the header is a toolbar with "Actions" and "Search Types" dropdowns, and a help icon. The main body of the dialog contains several input fields and dropdowns:

- User Name:** Text input field with an asterisk (*) placeholder.
- First Name:** Text input field with an asterisk (*) placeholder.
- Last Name:** Text input field with an asterisk (*) placeholder.
- Role:** Drop-down menu with an asterisk (*) placeholder.
- Company:** Drop-down menu showing "Company Name" as the current selection.
- Vault:** A section containing four radio buttons:
 - User Default (radio button)
 - Local (radio button)
 - Selected (radio button) followed by a text input field and a "...>" button.
 - All (radio button, currently selected)

At the bottom of the dialog are search controls:

- Limit to:** A dropdown menu set to "100" followed by "results".
- Paginate results:** A checkbox.
- Search:** A blue button with a magnifying glass icon.
- Cancel:** A red button with a white "X" icon.

emxUIDialog.css

To change how the dialog's top and bottom frames appear, edit the style sheet file emxUIDialog.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
body { }	Background Appearance	Background

ENOVA Microsoft Internet Explorer

Edit RFQ Details

Fields in red labels are required.

Name	94
RFQ Template	Test RFQ Temp atc 1
WorkSpace Folder	<input type="text"/> <input type="button" value="..."/> <input type="button" value="Clear"/>
Buyer Desk	<input type="text"/> <input type="button" value="..."/> <input type="button" value="Clear"/>
Comments	<input type="text"/>
Description	<input type="text"/>
Unit Price Formula	<input type="text"/> <input type="button" value="..."/>
<input checked="" type="button" value="Done"/> <input type="button" value="Cancel"/>	

Background appearance



emxUIForm.css

To change how the dialog's middle frame appears, edit the style sheet file emxUIForm.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
body { }	Background Appearance	Background
td.label { }	Default Label Appearance	Background, Font
td.requiredLabel { }	Required Label Appearance	Background, Font
td.inputField { }	Input Field Appearance	Background, Font
td.requiredNotice { }	Required Notice Appearance	Font

Edit RFQ Details

Fields in red letters are required.

Required Notice Appearance	Name: 94 RFQ Template: Test RFQ Temp ato 1 Workspace Folder: <input type="text"/> <input type="button" value="Clear"/>
Default Background Appearance	Buyer Desk: <input type="text"/> <input type="button" value="Clear"/>
Required Label Appearance	Comments: <input type="text"/>
Default Label Appearance	Description: <input type="text"/>
Input Field Appearance	Base Expression: <input type="text"/> Unit Price Formula: <input type="text"/>

Done **Cancel**

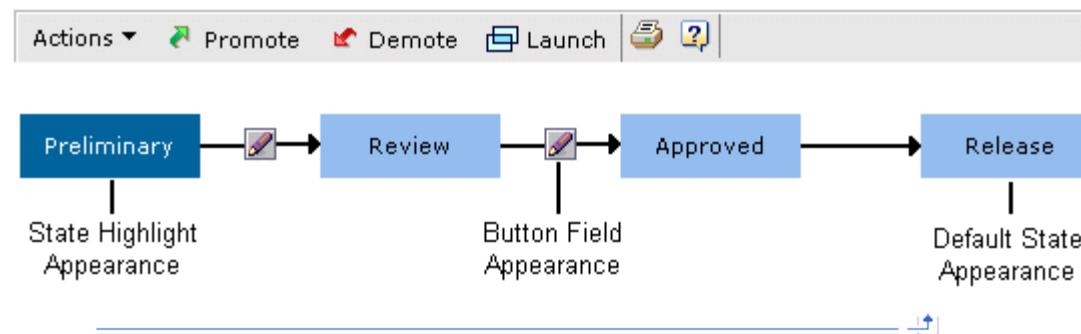
emxUILifeCycle.css

To change how the Lifecycle pages appear, edit the style sheet file emxUILifeCycle.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
td { }	Default Cell Appearance	Position
td.buttonField { }	Button Field Appearance	Background, Position
td.stateName { }	Default State Appearance	Font, Background, Position
td.stateNameHighlight { }	State Highlight Appearance	Background, Font, Position
td.shadow { }	State Shadow Appearance	Background

1 GB 2 DIMM rev 1: Lifecycle



emxUIList.css

To change how the table pages appear, edit the style sheet file emxUIList.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
th { }	Table Header Appearance	Background, Font, Position
th.sorted { }	Sorted Table Header Appearance	Background
th a { }	Table Header Link Appearance	Font
th.sub { }	Sub Table Header Appearance	Background, Font, Position
th.subSorted { }	Sorted Sub Table Header Appearance	Background
tr.odd { }	Odd Table Row Appearance	Background
tr.even { }	Even Table Row Appearance	Background

Tasks					
Actions					
Name	Action	Instructions	Due Date	Recipe	Approver
Task - 100	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Auxiliaries	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Back Seat	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Bearing	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Body	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Body Cover	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Notes and nuts	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Brake	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Brake Pedal	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Car	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Brake Pedal	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Car	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Brake Pedal	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Car	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith
Review					
Check sheet	Approve	Please review this file	March 6, 2001	BT-Info2903	Joe Smith

emxUIListPF.css

To change how the printer-friendly table pages appear, edit the style sheet file emxUIListPF.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
th { }	Table Header Appearance	Background, Font, Position
th.groupheader { }	Table Header Column Group Header	Background, Font, Position

Table Header Appearance		Table Header Column Group Header Appearance		
Part Families		Everything, Test Jan 10, 2006		
Name	Rev	Type	Description	State
123	-	Part Family	123	Active
PartFamily1	-	Part Family	PartFamily1	Active
PartFamily2	-	Part Family	PartFamily2	Active
PartFamily3	-	Part Family	PartFamily3	Inactive

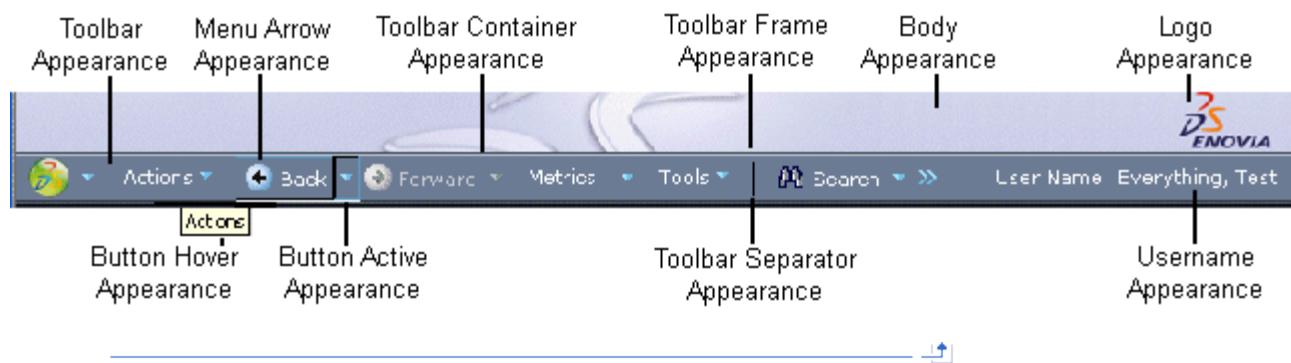
emxUINavigator.css

To change how the components of the main page appears, edit the style sheet file emxUINavigator.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
body.navigator { }	Body Appearance	Background
#divShrunkLogo { }	Logo Appearance	Position
#divShrunk { }	Header Appearance	Position
#elmUsername { }	Username Appearance	Font, Position
#content { }	Content Window Appearance	Position
.toolbar-row { }	Toolbar Row Appearance	Background, Borders
div.toolbar { }	Toolbar Appearance	Background, Borders, Position
div.toolbar-container { }	Toolbar Container Appearance	Background, Borders
div.toolbar-frame { }	Toolbar Container Frame Appearance	Background, Borders
.icon-button, .text-button,.icon-and-text-button,.overflow-button { }	Button Appearance	Background, Borders, Font
td.menu-arrow { }	Menu Arrow Appearance	Background
td.combo-button { }	Combo Button Appearance	Position
.text-button { }	Text Button Appearance	Position
.overflow-button { }	Overflow Button Appearance	Background
.icon-button { }	Icon Button Appearance	Position
.icon-and-text-button { }	Icon and Text Button Appearance	Background, Position
.icon-button span.down-arrow , .text-button span.down-arrow , .icon-and-text-button span.down-arrow{ }	Button Arrow Appearance	Background, Position
.button-hover, td.button-hover { }	Button Hover Appearance	Background, Borders

.button-active,.menu-button-active,td.button-active { }	Button Active Appearance	Background, Borders
.button-disabled{ }	Button Disabled Appearance	Background
.separator{ }	Toolbar Separator Appearance	Borders



emxUIProperties.css

To change how the Properties pages appear, edit the style sheet file emxUIProperties.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
td.label { }	Default Label Appearance	Background, Font
td.field { }	Display Field Appearance	Background

Default Label Appearance Display Field Appearance

123 : Properties

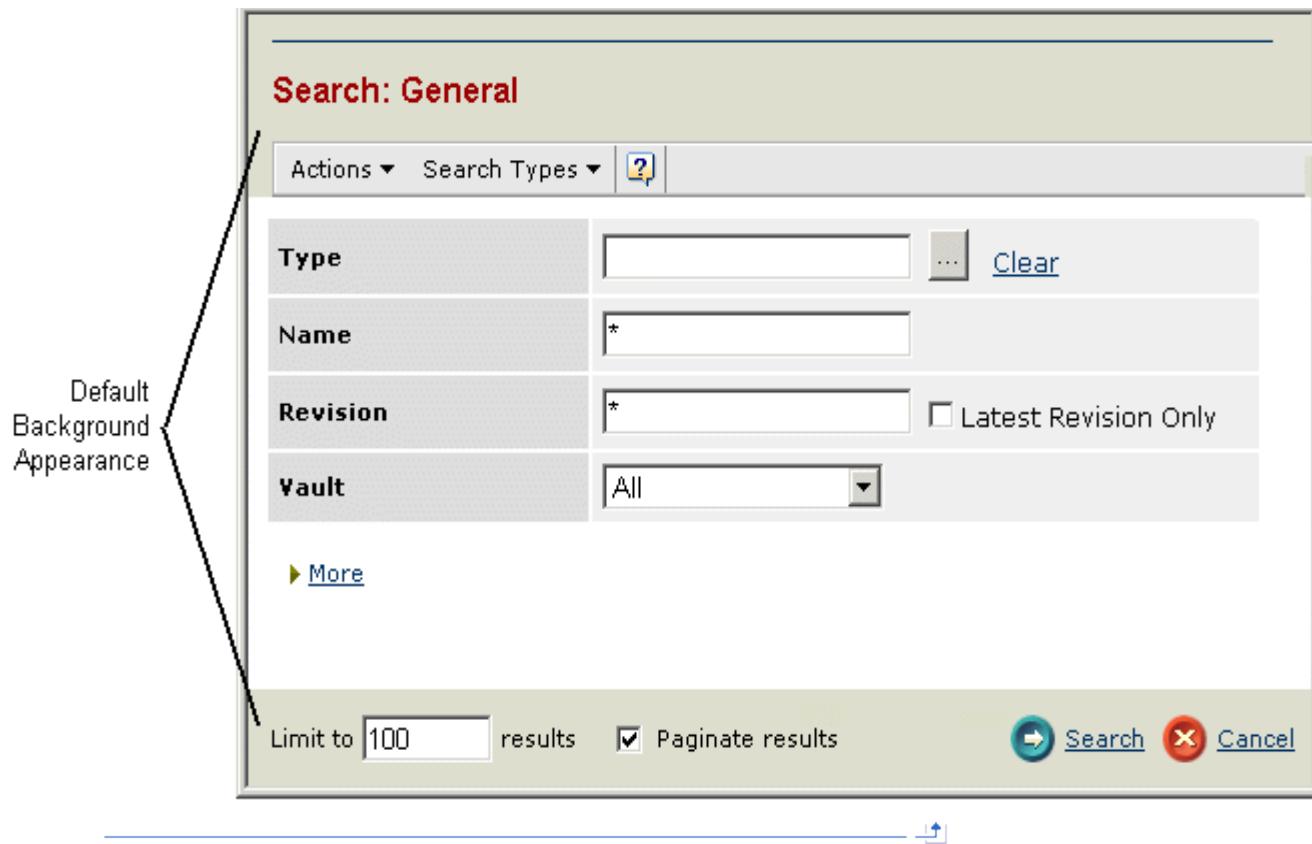
Edit Details	Subscribe	
Description	123	
State	Active	
Owner	ProductManager, Test	
Originator	ProductManager, Test	
Originated	Dec 6, 2005	
Modified	Dec 15, 2005	
Name Generator	On	
Sequence Pattern	0000	
Prefix Pattern		
Base Number		
Suffix Pattern		
Pattern Separator	-	
Classification Path	123	

emxUISearch.css

To change how the list appears, edit the style sheet file emxUISearch.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
body { }	Default Background Appearance	Background



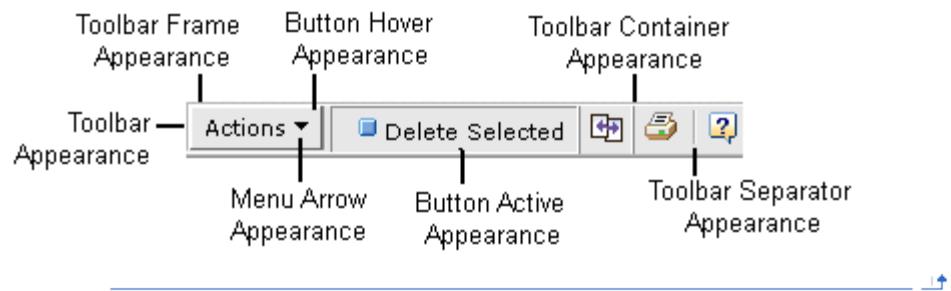
emxUIToolbar.css

To change how the dialog's toolbar appears, edit the style sheet file emxUIToolbar.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
div.toolbar { }	Toolbar Appearance	Background, Borders, Position
div.toolbar-container { }	Toolbar Container Appearance	Background, Borders
div.toolbar-frame { }	Toolbar Container Frame Appearance	Background, Borders
.icon-button, .text-button,.icon-and-text-button,.overflow-button { }	Button Appearance	Background, Borders, Font
td.menu-arrow { }	Menu Arrow Appearance	Background
td.combo-button { }	Combo Button Appearance	Position
.text-button { }	Text Button Appearance	Position
.overflow-button { }	Overflow Button Appearance	Background
.icon-button { }	Icon Button Appearance	Position
.icon-and-text-button { }	Icon and Text Button Appearance	Background, Position
.icon-button span.down-arrow, .text-button span.down-arrow ,.icon-and-text-button span.down-arrow{ }	Button Arrow Appearance	Background, Position

.button-hover, td.button-hover { }	Button Hover Appearance	Background, Borders
.button-active,.menu-button-active,td.button-active { }	Button Active Appearance	Background, Borders
.button-disabled{ }	Button Disabled Appearance	Background
.separator{ }	Toolbar Separator Appearance	Borders



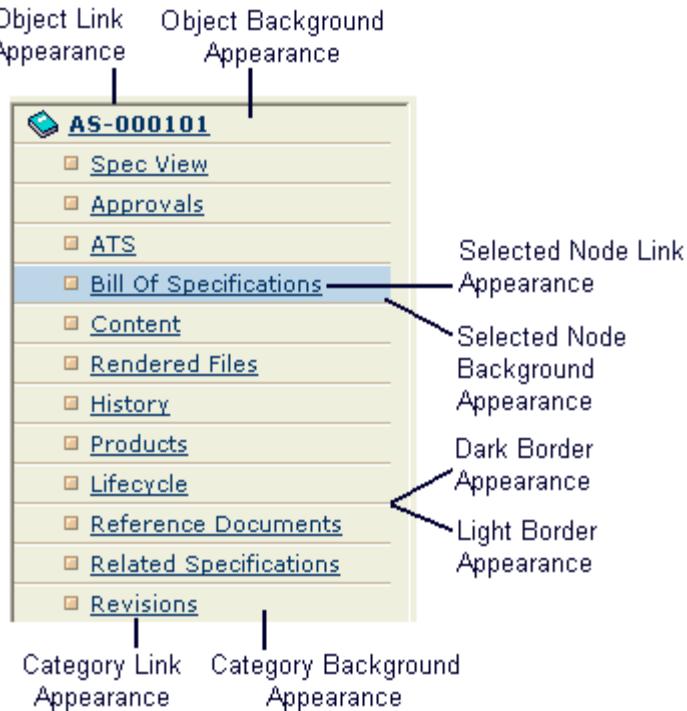
emxUITree.css

Avoid increasing the font sizes for text in the navigation trees because it will likely break the images or make the tree look distorted. To change how the navigation trees appear, edit the style sheet file emxUITree.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
body { }	Background Appearance	Background
* { }	Default Font Appearance	Font
a { }	Default Link Appearance	Font
td.object { }	Object Background Appearance	Background
td.object a { }	Object Link Appearance	Font
td.category { }	Category Background Appearance	Background
td.category a { }	Category Link Appearance	Font
tr.selected { }	Selected Node Background Appearance	Background
tr.selected a { }	Selected Node Link Appearance	Font
tr.lightBorder { }	Light Border Appearance	Background
tr.darkBorder { }	Dark Border Appearance	Background

The following image shows how each CSS style affects a details tree.



emxUIWizard.css

To change how the wizard pages appear, edit the style sheet file emxUIWizard.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
body { }	Default Background Appearance	Background

The screenshot shows the "Step 1 of 4: Specify Details" wizard page. The page has a header with "Add Content", "Upload External...", "Remove Selected", and a help icon. Below the header, a message says "Fields in red italics are required". The form consists of three rows of input fields:

Name	<input type="text"/>	<input checked="" type="checkbox"/> Autoname
Template	<input type="text"/>	<input type="button"/> Clear
Description	<input type="text"/>	

At the bottom are "Next" and "Cancel" buttons.

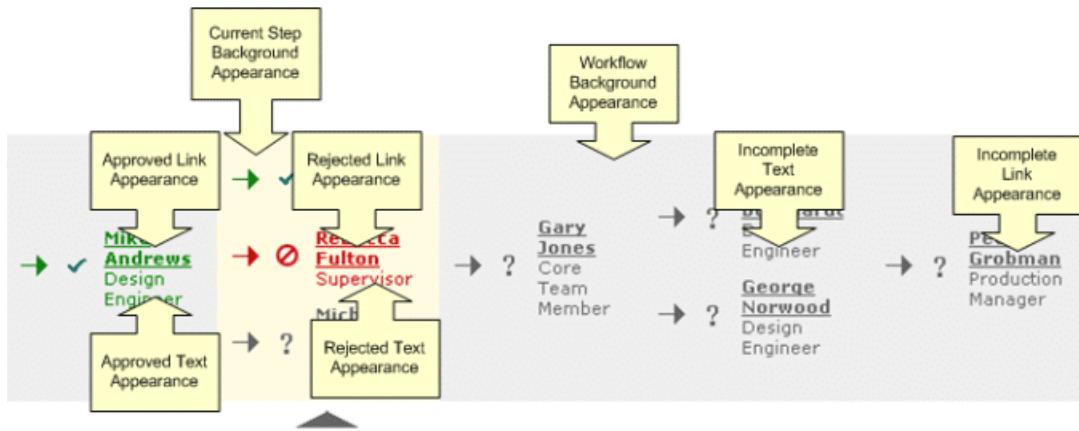
A callout line points from the text "Default Background Appearance" to the background color of the main content area of the wizard page.

emxUIWorkflow.css

To change how the list appears, edit the style sheet file emxUIWorkflow.css.

The style sheet contains the following style definitions:

Definition	Name	Categories
tr.workflow { }	Workflow Background Appearance	Background
td.current { }	Current Step Background Appearance	Background
td.incomplete { }	Incomplete Text Appearance	Background, Font, Position
td.approved { }	Approved Text Appearance	Background, Font, Position
td.rejected { }	Rejected Text Appearance	Background
td.incomplete a { }	Incomplete Link Appearance	Background, Font
td.approved a { }	Approved Link Appearance	Background, Font
td.rejected a { }	Rejected Link Appearance	Background, Font



Main Navigator Page

The main Navigator page contains the dynamic user interface components common to all applications and all commands: the banner and global toolbar, and content frame (where the table and form pages display). This page is called for all dynamic UI applications and is constructed by emxNavigator.jsp.

In this section:

- [About the Content Page for emxNavigator.jsp](#)
- [Configuring the Main Navigator Page](#)
- [Configuring the Default Home Page for the Content Frame](#)
- [URL Parameters Accepted by emxNavigator.jsp](#)

About the Content Page for emxNavigator.jsp

If you call more than one parameter or properties file setting to define the content page for emxNavigator.jsp, the system uses the precedence order explained in this section to determine which page to load.

1. Any parameter included with emxNavigator.jsp that is specified in a URL link.

Any parameter specified in an external URL link (for example, in an email or Web site) overrides all other settings.

2. User's Home page preference setting.

The user's Home page preference overrides the content page settings in emxSystem.properties. You can use the properties to configure the content page for when users first log in, but users can override this by choosing a preferred Home page.

3. Any parameter included with emxNavigator.jsp that is specified in the eServiceSuiteFramework.URL property in emxSystem.properties. If more than one parameter for the content page is specified in the property, the system uses this precedence order:

- a. ContentPage parameter
- b. MenuName/CommandName parameters that call the URL associated with the command object
- c. objectId parameter for tree for object

4. `emxNavigator.Home.ContentPage` property specified in emxSystem.properties.

If none of these parameters are passed or any error conditions occur because of an incorrect configuration, the default home page (emxDefaultNavigatorContentPage.jsp) is displayed in the content frame.

Configuring the Main Navigator Page

You can configure how the Navigator page looks when a user first logs in or when the page is called externally from an email message or other Web page by defining the default page to display in the content frame.

The content frame is the main work area of the Navigator page, where the table and form pages display. It is called the Home page within the user interface. Users can use the Preferences tool to choose their own Home page preference, which overrides any page that you specify.

1. When calling emxNavigator.jsp, pass these URL parameters and appropriate values:

- CommandName
- ContentPage
- MenuName
- mode
- objectId
- portal

See [URL Parameters Accepted by emxNavigator.jsp](#) for details.

2. To define the default configuration for initial user log in, enter a value for the `eServiceSuiteFramework.URL` property in `emxSystem.properties` file. See the *Live Collaboration Administrator's Guide*.
3. To implement a configuration when calling emxNavigator.jsp from a link in a Web page or email message, add the appropriate parameter to the emxNavigator.jsp URL for the link.
For information on accessing emxNavigator.jsp through an external link, see [Accessing Applications Externally](#).
4. Specify the default page for the content frame. See [Configuring the Default Home Page for the Content Frame](#).

Configuring the Default Home Page for the Content Frame

There are a number of ways you can specify the page that displays in the content frame of the Navigator window when users log in. The content frame is referred to as the Home page in the user interface. You can use one of these methods:

This task shows you how to:

- [Pass the ContentPage Parameter to emxNavigator.jsp](#)
- [Pass the CommandName and MenuName Parameters to emxNavigator.jsp](#)
- [Pass an objectId to emxNavigator.jsp](#)
- [Define a Home Page in emxSystem.properties](#)

Pass the ContentPage Parameter to emxNavigator.jsp

The value for the ContentPage can be the URL for any JSP, plus any needed URL parameters.

- Define the value for the ContentPage URL parameter to pass to emxNavigator.jsp.

The value for the ContentPage can be the URL for any JSP, plus any needed URL parameters. For example:

```
emxNavigator.jsp?ContentPage=.../engineeringcentral/  
emxMyTasksSummaryFS.jsp
```

Where .../engineeringcentral/emxMyTasksSummaryFS.jsp is the URL for the Tasks page within Engineering Central.

The URL assigned to the ContentPage parameter can contain only one parameter/value pair as part of the URL. Any additional parameters are ignored.



Pass the CommandName and MenuName Parameters to emxNavigator.jsp

The values for the CommandName and MenuName parameters combine to define which content page should be loaded.

- Define the value for both the CommandName and MenuName URL parameters to pass to emxNavigator.jsp.

These parameters let you specify the name of a menu and command object pair to display in the content frame. The system displays the URL associated with the command object attached to the menu. The parameters are:

- MenuName. Gets the name of the menu object for the submenu that has the command object attached. The links in the Actions submenus cannot be used for display in the content frame.
- CommandName. Gets the name of the administrative command object from which the href URL is to be obtained.

For example:

```
emxNavigator.jsp?MenuName=TMCMYDesk&CommandName=TMCRoutesMyDesk
```

Where TMCMYDesk is the name of the menu object for the Team Central submenu in the DS (formerly My Desk) menu and TMCRoutesMyDesk is the name of the command object for the Routes link. This command object contains an href parameter that points to the JSP for the Routes page.



Pass an objectId to emxNavigator.jsp

You can pass an objectId to load the tree for that specific object.

- Define a value for the objectId URL parameter to pass to emxNavigator.jsp.

For example:

```
emxNavigator.jsp?objectId= 29547.17050.1266.35700
```

Where 29547.17050.1266.35700 is the ID for the business object for which the default page for that object.



Define a Home Page in emxSystem.properties

You can define a value for the emxNavigator.Home.ContentPage property in the emxSystem.properties file to define which page to open.

1. Open the emxSystem.properties file for editing.
2. Locate this line:

```
emxNavigator.Home.ContentPage =  
emxDefaultNavigatorContentPage.jsp
```

By default this property is set to the default home page, which is the welcome page or emxDefaultNavigatorContentPage.jsp.

- 3.** Set the value for this property to the name of the jsp page you want to use as the system default.
- 4.** If needed, continue editing the emxSystem.properties file. If finished, save the file.
- 5.** When finished making all properties file edits, restart the application server. If using a J2EE implementation, run the warutil and deploy files as usual.

URL Parameters Accepted by emxNavigator.jsp

This table lists the parameters that can be passed to emxNavigator.jsp.

Parameter	Description	Accepted Input Values
CommandName	<p>Use along with the MenuName parameter to define the page to display in the Content frame of the Navigator page when users log in. The MenuName parameter defines the name of a menu object that represents a submenu and the CommandName parameter defines the name of a command object where the href URL can be obtained.</p> <p>Note that the links in the Actions tab submenus cannot be used for display in the content frame.</p>	<p>The name of a command object that is assigned to a menu object that represents an application submenu on the My Desk menu.</p> <p>For example: <code>emxNavigator.jsp?MenuName=TMCMyDesk&CommandName= TMCRoutesMyDesk</code></p> <p>Where TMCMyDesk is the name of the menu object for the Team Central submenu and TMCRoutesMyDesk is the name of the command object for the Routes link. This command object contains an href parameter that points to the JSP for the Routes page.</p>
ContentPage	Defines the JSP to appear in the Content frame (called the Home page in the user interface) when you call the Navigator page.	<p>Any JSP plus parameters:</p> <p><code>emxNavigator.jsp?</code> <code>ContentPage=../engineeringcentral/emxengchginboxFrameset.jsp</code></p> <p>Currently, the URL assigned to the ContentPage parameter can contain only one parameter/value pair as part of the URL. Any additional parameters are ignored.</p>
MenuName	<p>Defines the application submenu to expand when you call the Navigator page. If you don't specify a submenu, the first menu in the My Desk is expanded.</p> <p>Can also be used along with the CommandName parameter to define the page to appear in the Content frame on login.</p>	<p>The name of a menu object that represents an application submenu.</p> <p>For example: <code>emxNavigator.jsp?MenuName=TMCMyDesk</code></p>
mode	<p>Used when calling the Navigator page from an external link on a Web site, ENOVIA portal page, or email message. Defines whether the Navigator window contains the banner and global as it normally does when using the applications.</p>	<ul style="list-style-type: none"> • Menu. The Navigator page contains all standard elements. • Tree. The Navigator page contains only the tree frame and the content frame. <p>For example: <code>emxNavigator.jsp?</code> <code>mode=Menu&ContentPage=../sourcingcentral/emxBuyerDeskTable.jsp</code></p>
objectId	Defines the business object to display when you call the Navigator window.	<p>Valid business object ID. 2233.5567.2323.4678</p>
portal	Used when calling the Navigator page from an external link to ensure the Login page does not replace the frame that contains the link.	<ul style="list-style-type: none"> • true. Turns the ENOVIA portal mode on to ensure the Login page and Navigator page have no effect on the page that contains the URL link to the Navigator page. • false. Turns the portal mode off. If the parameter isn't specified, the portal mode is turned off.

./Apps/Framework/VERSION/commonUI/emxNavigator.jsp?portal=true

Adding a Custom Application

This procedure describes the steps you need to follow to integrate a custom application into the dynamic user interface. The custom application appears in the My Desk menu along with ENOVIA products.

1. Create a directory for your application's JSPs within the ematrix directory. This directory is under the staging directory within the RMI installation directory. This directory is equivalent to the subdirectories for the ENOVIA products, such as ematrix/engineeringcentral. Place all your JSPs in this directory.
2. Create an images directory under the subdirectory you created in Step 1. Place all image files for your application in this directory.

You must run the warutil and deploy the archive file when you are done adding files for the application.

3. List your custom application in the DisplayedSuites property in emxSystem.properties. The name you give it here is the name you should use when adding the properties for the suite in Step 4.

Custom applications cannot have the same name as an existing application and they cannot have the same name as an existing application without the "eServiceSuite" prefix. For example, the name for Engineering Central is eServiceSuiteEngineeringCentral so you cannot have a custom application with the name eServiceSuiteEngineeringCentral or EngineeringCentral.

```
eServiceSuites.DisplayedSuites =  
eServiceSuiteEngineeringCentral, \  
    eServiceSuiteTeamCentral, \\  
        CustomCentral \
```

4. Add a set of properties to emxSystem.properties to tell the system the name of your application directory, properties files, and other important information. Each ENOVIA product has a set of properties and the set for a custom application called Custom Central is shown below. Create an equivalent set for your application and change the property names and the values to reflect your application. The property names should match the name used in the DisplayedSuites property. For a description of the properties, see the *Live Collaboration Administrator's Guide*.

```
# Custom Central  
CustomCentral.URL = common/emxNavigator.jsp  
CustomCentral.Directory = customcentral  
CustomCentral.ApplicationPropertyFile =  
emxCustomCentral.properties  
# ApplicationStartPage is within 'Directory' specified  
CustomCentral.ApplicationStartPage = emxCustomFrames.jsp  
CustomCentral.StringResourceFileDialog =  
emxCustomCentralStringResource  
CustomCentral.PropertyFileAlias=emxCustomCentralProperties
```

5. Define a submenu for the application. Creating a submenu for the application effectively registers it. The submenu will contain all top-level commands for your application. For instructions on creating menus and commands, see [About Building Menus](#).
6. Register new schema and configure existing schema as needed using the instructions in "Configuring the Schema" in the *Live Collaboration Administrator's Guide*.

If you add a custom type, you can specify the icon to use for the type's root tree node using the emxFramework.smallIcon.TYPE_SYMBOLIC_NAME property in emxSystem.properties. For more information, see [About the Icon for a Type's Tree](#).

7. Define additional and configure existing trees, tables, forms, and menus as needed.

Migration to the New UI

The new UI requires several changes to be made to custom pages. This section includes information and procedures to help developers migrate custom pages to the new UI.

In this section:

- [About the New User Interface \(UI\)](#)
- [About the FS Page Component](#)
- [Updating Framesets and DOCTYPE References](#)
- [Converting JavaScript Frame References](#)
- [Converting Pages to the New UI Framework](#)
- [Restructuring Menus](#)
- [Implementing Slide-in Dialogs and Edit Dialogs](#)
- [Removing Inline Style Definitions and Invalid HTML Elements](#)
- [Enabling Automatic Type Ahead](#)
- [Updating PowerView Channel Heights](#)
- [Table-to-Structure Browser Conversion](#)

About the New User Interface (UI)

Release V6R2012 includes a simplified and improved user interface. This section describes the new features of the UI.

- [New Architecture](#)
- [Checklist](#)

The new UI includes these features:

- CSS changes for the look-and-feel of the ENOVIA products
- Mega-menus that allow a user to click once to see ALL available menu options and then one more click to select the needed menu
- Pre-defined popup dialogs
- More room for data by moving the category list to a menu
- Breadcrumb trail
- Table components migrated to structure browser components
- Improved PowerViews (emxPortal.jsp)
- Thumbnails in results
- Consolidated Image Viewer

The new UI also reduces the number of popup windows:

- Slide in dialogs
- Type-ahead form fields in place of popup search pages

This screen shows an example of the new look-and-feel for ENOVIA products.

The screenshot displays the ENOVIA BOM PowerView interface. At the top, there's a toolbar with various icons. Below it is a breadcrumb trail: 'Car 1 - BOM PowerView > Tabbed 1 (1.0) - Home'. The main area has a title bar 'Welcome: Everything, Icat' and a navigation bar with tabs: 'Categories', 'Bill of Materials', 'Where Used', 'Spare Parts', and 'Markups'. The 'Bill of Materials' tab is selected, showing the heading 'Car rev 1: Engineering Bill of Materials'. Below this is a toolbar with buttons for 'Actions', 'Reports', 'Edit All', '3DVIA Viewer', 'Compliance', 'Launch', 'Expand', 'Filter', 'Add To', 'Markup', 'Undo', and 'Search within C...'. A table follows, with columns: Name, Type, Rev, Policy, F/N, Ref Dcs, Compon, Description, State, and Qty. The table lists 10 objects, including 'Car', 'Engine', 'Body', 'Fuel Tank', 'Seal', 'Body Cover', 'Mirror', 'Wheel', and 'Dash Board'. Each row provides details like Project ID, Revision, and State. The bottom of the table shows a footer with '10 objects'.

This sample includes the breadcrumb trail (Car 1: BOM PowerView line). Users can click on any page listed to go directly to that page.

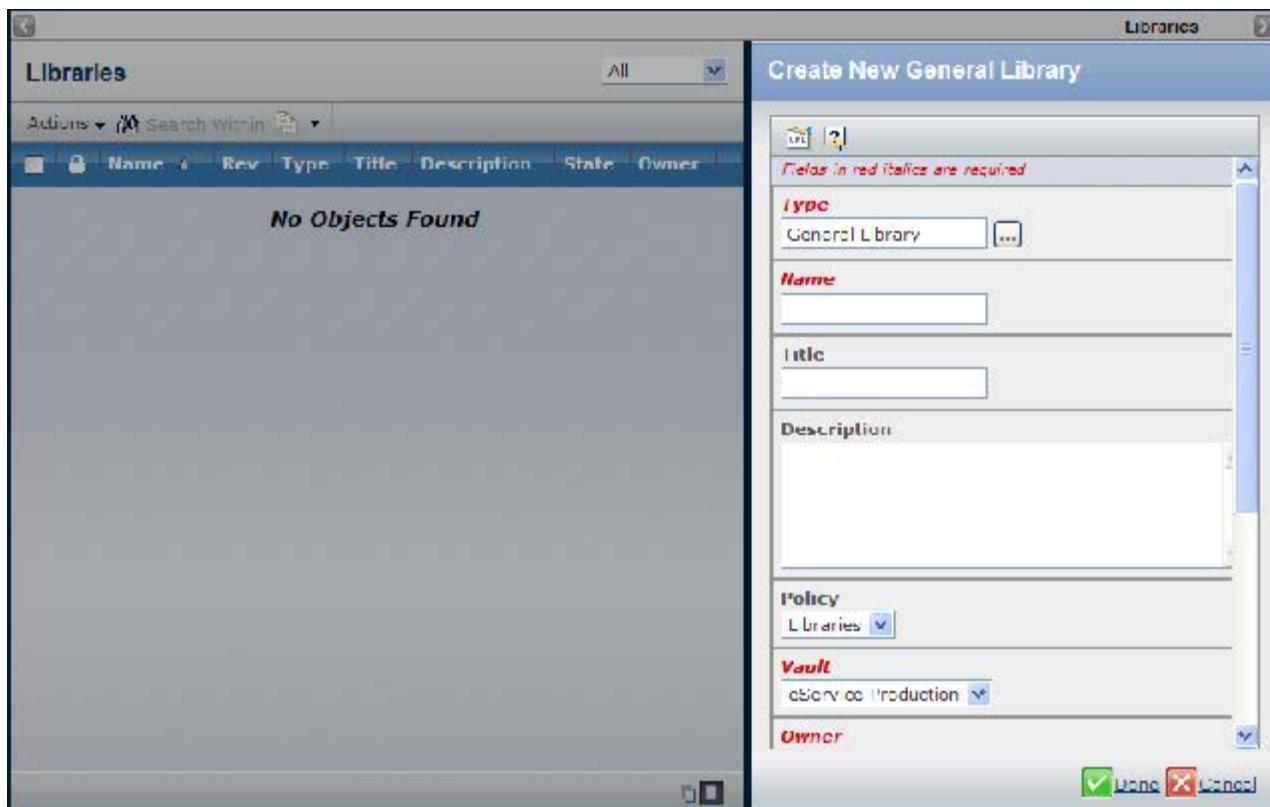
This sample also shows the Categories menu, which is comprised of the contents of the tree defined for the context object type. By moving the category list to a menu, the structure navigator has more room to expand. On pages that do not have a structure navigator, there is more room to display data. The structure navigator displays the hierarchy of structured items such as Folders, Parts (as shown above), and any other structured object.

This screen shows the mega menus:

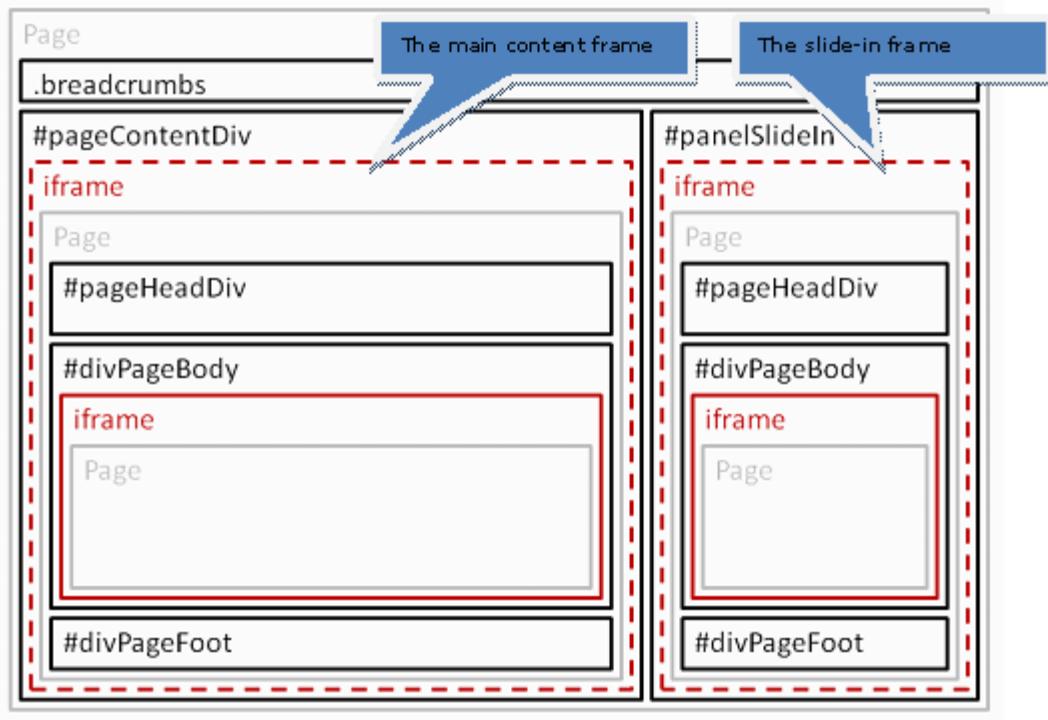


A user clicks  to open the menu, and then the item they need. Only end-items are clickable, the headings (such as Engineering or Supplier) provide organizational structure to help the user find the needed menu. To select a menu, the user only needs to click twice: once to open the menu and once on the needed option. This style menu is also used for the Actions, Tools, and Search menus from the global toolbar, and Actions menus on page toolbars.

Slide-in dialogs are used to replace pop-up windows. Previously, when forms are popped up from various windows, a user can have several ENOVIA windows open at the same time. These windows can become confusing. With the slide-in window, the original window becomes disabled until the user closes the slide-in window.



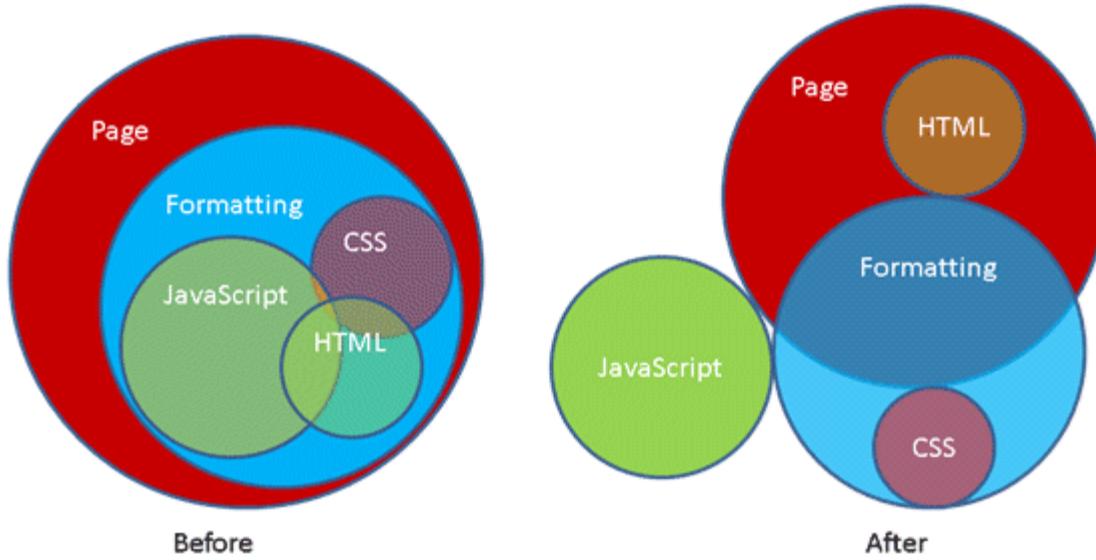
This diagram shows the structural component of a slide-in frame next to the main content frame:



The Image Viewer can now show 2D and 3D images in the same window.

New Architecture

The New UI adopts industry standard best practices for composing a page to present to the user. To do this, competing approaches to rendering elements on a page have been minimized. For example, CSS is preferred over the use of HTML to format page elements. In addition, the complexity of pages has been reduced by separating out elements into well-defined locations. For example, in-line CSS has been removed, placing all CSS into separate files that can be more easily maintained by developers. This type of separation is illustrated in the diagram below.



This architecture makes it easier to maintain the code base and provides greater control over how pages are rendered. In addition, a greater percentage of the pages have been moved into configurable components. This makes it easier to migrate pages to successive releases without needing to modify those pages to take advantage of new features or to minimize any deficiencies introduced. The definition of the page remains unchanged, but how that page is composed and rendered may change without altering the page definitions.

Checklist

This checklist shows the required and optional changes you need to make to support the V6R2012 user interface.

Task	See for Instructions	Optional / Required
------	----------------------	---------------------

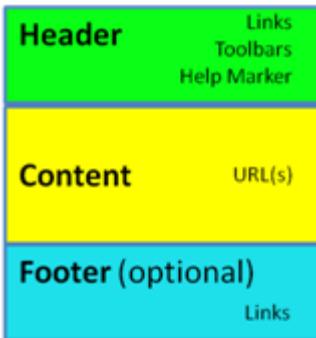
Remove Frameset HTML tags	Updating Framesets and DOCTYPE References	Required
Fix DOCTYPE references	Updating Framesets and DOCTYPE References	Required
Convert JavaScript frame references in component pages	Converting Pages to the New UI Framework	Required
Migrate custom JSP pages	Converting Pages to the New UI Framework.	Required
Enable type ahead for Person, Organization, and Type fields	Enabling Automatic Type Ahead.	Optional
Convert popup dialogs to slidein Windows	Implementing Slide-in Dialogs and Edit Dialogs	Optional
Update FS Page references	Converting Pages to the New UI Framework	Optional
Update Content Page references	Implementing Slide-in Dialogs and Edit Dialogs	Optional
Remove inline CSS styles.	Removing Inline Style Definitions and Invalid HTML Elements	Optional
Remove invalid HTML tags.	Removing Inline Style Definitions and Invalid HTML Elements	Optional
Specify pre-defined sizes for popup dialogs.	Removing Inline Style Definitions and Invalid HTML Elements	Optional
Rearrange menus for mega menus.	Restructuring Menus	Optional

About the FS Page Component

Custom pages that used <frameset> tags can either be changed to use configurable components, or to use the FS Page component described in this topic. This topic describes the page structure, and includes sample HTML code using the FS Page component.

Pages developed for prior releases often used the <frameset> tag on custom pages, and within frames within framesets. Pages in the new UI should not use <frameset> tags. Developers can use the FS Page component to migrate custom pages.

The FS Page component uses the Document Object Model (DOM) to control the placement of page sections and a Java Bean generates the final page. It allows you to provide the parameters for the header, and if needed, the footer. The content area of the page is placed in an iframe, to allow users to provide one or more URLs for the Content section. This graphic shows the FS Page structure.



Any previously written code for the content frame under the old UI can be used within the iframe for the new content section with minimal modification.

This HTML code shows a DOM page using the FS Page component:

```
<%@include file="../emxUIFramesetUtil.inc"%>
<%@include file="../common/emxNavigatorInclude.inc"%>

<%
    framesetObject fs = new framesetObject();

    fs.setDirectory(appDirectory);
    fs.setSubmitMethod(request.getMethod());

    String initSource = emxGetParameter(request,"initSource");
    if (initSource == null){
        initSource = "";
    }

    String jsTreeID = emxGetParameter(request,"jsTreeID");
    String suiteKey = emxGetParameter(request,"suiteKey");
    String objectName = emxGetParameter(request,"objectName");

    // ----- Do Not Edit Above -----

    // Add Parameters Below

    // Specify URL to come in middle of frameset
    StringBuffer contentURLBuf = new StringBuffer(80);
    contentURLBuf.append("Themiddlecontentpage.jsp");

    // add these parameters to each content URL, and any others the App needs
    contentURLBuf.append("?suiteKey=");
    contentURLBuf.append("&initSource=");
    contentURLBuf.append(initSource);
    contentURLBuf.append("&jsTreeID=");
    contentURLBuf.append(jsTreeID);

    String contentURL=FrameworkUtil.encodeHref(request,contentURLBuf.toString());

    // This is the string resource key for the Page Heading - Internationalized
    String PageHeading = "emxFramework.Command.CreateCollection";

    // Marker to pass into Help Pages
    // icon launches new window with help frameset inside
    String HelpMarker = "emxhelpcollectioncreate";

    fs.initFrameset(PageHeading,
                    HelpMarker,
                    contentURL,
                    false,
                    true,
                    false,
```

```

        false);

// this method sets the string resource key. Generally this should be the app specific string resource
file.
fs.setStringResourceFile("emxFrameworkStringResource");

fs.setCategoryTree((String)emxGetParameter(request,"categoryTreeNode"));
fs.setToolbar("toolbarname");

// Narrow this list and add access checking
String roleList = "role_GlobalUser";

// This is the sample method to create a button in the toolbar in the header
fs.createHeaderLink("emxFramework.Common.DeleteSelected",
                    "deleteObject()", 
                    roleList,
                    false,
                    true,
                    "emxUIButtonDelete.gif",
                    0);

// This is the sample method to create a button in the Footer page
fs.createFooterLink("emxFramework.Common.Done",
                    "doneMethod()", 
                    roleList,
                    false,
                    true,
                    "emxUIButtonDone.gif",
                    0);

// This is the sample method to create a button in the Footer page
fs.createFooterLink("emxFramework.Button.Cancel",
                    "parent.window.close()", 
                    roleList,
                    false,
                    true,
                    "emxUIButtonCancel.gif",
                    0);

// ----- Do Not Edit Below -----
fs.writePage(out);
%>

```

In the above code, these sections correspond to the indicated parts of the screen capture:

Type	Issue
Description	
Reported Against	
Escalation Required	<input type="radio"/> Yes <input checked="" type="radio"/> No
Estimated Start	
Estimated Finish	
Priority	Low
Co-owners	
Problem Type	Not Determined

1. The `String PageHeading` line specifies the page heading (internationalized).
2. The `String HelpMarker` line specifies the help marker.
3. These are used to add links to the toolbar. Use one or the other, but not both:
 - The `fs.setToolbar` line defines the toolbar. this line can be omitted if the toolbar contains no items
 - The `fs.createHeaderLink` lin adds links to the toolbar
4. The `// Add Parameters Below` section specifies the content or body portion of the page
5. The `fs.createFooterLink` lines specify links in the page footer.

Updating Framesets and DOCTYPE References

To allow custom pages to run under the new UI, you must remove <frameset> HTML tags and replace them with <DIV> tags. DOCTYPE references also need to be removed from all custom pages.

1. Remove all HTML <frameset> tags from all custom pages. Framesets have been removed from Table, Form Component, and the FS Pages.

All frameset pages need to be replaced with DOM-based layouts (see [About the FS Page Component](#)). In most custom pages, applications refer to the body frame for data. The new UI retains this element by replacing the body frame with an iframe. The header and any footer page parts are moved into separate <DIV> elements.

The new UI uses HTML5 DOCTYPE which does not support the frameset tag.

2. DOCTYPE tags must be removed from all of your application pages and DOCTYPE html tags must be removed from custom JSP pages.

The DOCTYPE tag, or document type declaration, is used by web browsers as a cue on how to parse and render a page. The V6R2012 release automatically inserts the DOCTYPE tag at the beginning of the page. If you have a separate doctype declaration, it may interfere with this automatic addition of the DOCTYPE.

The first line in any generated HTML file must be <html>. In addition, the HTML must be well-formed, valid HTML code. All tags must be nested correctly, and only HTML tags can be used.

3. Replace all references to `emxNavigatorInclude.inc` with references to `emxNavigatorNoDocTypeInclude.inc`.

Replace all references to `emxUICommonAppInclude.jsp` with references to `emxUICommonAppNoDocTypeInclude.jsp`.

These files contain checks to ensure the user is currently logged in along with an HTML 5 doctype tag. This directive may not be needed for pages such as PreProcess JSPs that are included as part of other JSPs. These JSPs should not have doctype as the main page already has the doctype.

The new UI adds the <!DOCTYPE html> directive to these pages which is a necessary as part of this NewUI.

Converting JavaScript Frame References

You need to convert all JavaScript frame references in JSP pages to move headers and footers to <DIV> elements and body content and hidden frames to an iframe.

1. Convert JavaScript frame references in configurable component pages to move headers and footers to <DIV> elements and body content and hidden frames to an iframe.

You need to make these changes for the JSPs listed in this table. This table lists frames that are now contained in iFrames. Some frames that were previously visible are now hidden. These changes provided backward compatibility and allow configuration setting changes that support showing selected elements with the UI used in the prior release.

Category	Frame Name	Visible
Navigator	content	Yes
	hiddenFrame	No
	integrationsFrame	No
	appletFrame	No
	ClipboardCollection	No
View Form	formViewDisplay	Yes
	formViewHidden	No
Edit Form	formEditDisplay	Yes
	formEditHidden	No
Table	listDisplay	Yes
	listHidden	No
	postHidden	No
FS Pages	pagecontent	Yes
	pagehidden	No
	jpcharfooter	No
Structure Navigator	emxUIStructureTree	Yes
	detailsDisplay	Yes
	hiddenTreeContentFrame	No
Custom Search	searchContent	Yes
	searchView	Yes
	searchHidden	No

2. Modify all references to header and footer frames that use an index (such as "frames[0]" or "frames['framename']"). ENOVIA recommends that you use the findFrame method. This example snippet of JavaScript shows how to change the references:

Before:

```
top.frames['framename'].document.location.href
```

After:

```
var frameobj = findFrame(top,'framename');
frameobj.document.location.href
```

3. Use the methods `turnOnProgress()` and `turnOffProgress()` to control the progress indicator shown in the header. Inline code for controlling the display of the progress indicator should be discontinued.

Converting Pages to the New UI Framework

To convert existing application pages to the new UI page framework, you can either convert the page to use a configurable component, convert the page to use the FS Page component, or modify the custom JSP page.

Which approach works for the pages in your application may vary based upon the content of the pages. To ensure the best compatibility with the new UI and for future compatibility issues, ENOVIA recommends you convert to configurable components where possible, and only modify the custom JSP pages if absolutely necessary.

This task shows you how to:

- [Convert to Configurable Component](#)
- [Convert to FS Page Component](#)
- [Modify Custom JSP Page](#)

Convert to Configurable Component

This method is the preferred choice for converting existing pages. In future releases, the underlying code for the component may be updated and the application developer will not need to modify their code to take advantage of those improvements.

- Refer to the sections of this guide that deal with the page type being converted:

- [Forms](#)
- [Structure Browser](#)
- [Tables](#)
- [Generic Create Form](#)
- [Searches](#)

The configurable components automatically generate the correct formatting of the pages and all other functionality associated with the new UI.



Convert to FS Page Component

This method wraps a header and footer around the content, but does not make other necessary changes to the page content.

The new UI moves away from using the html <frameset> tag on custom pages, along with pages for each of the frames used with the frameset. If you cannot convert a page to use a configurable component, ENOVIA recommends the use of the FS Page component to convert custom pages into the new UI. The FS Page component allows key parts of the existing page definitions within FS Page. Typically, a custom page would consist of a page for each frame and the <frameset> tag to pull together the header, content, and footer page sections.

- Migrate Custom JSP pages to an FS Page Component.

Use the Document Object Model (DOM) to control the placement of page sections. Provide the parameters for the header, and if needed, the footer. Place the content area of the page in an iframe to allow one or more URLs to use in the Content section.

Previously-written code for the content frame under the old UI may be provided for use within the iframe for the new content section with a minimal number of modifications.

See [About the FS Page Component](#) for code samples.



Modify Custom JSP Page

This method may require updates when a new release of the software is delivered.

Restructuring Menus

The new UI includes a new menu structuring mechanism, sometimes referred to as mega menus. Instead of combining drop-down menus and pull-right menus at various levels, the new menus present an indented structure of the entire menu system. The user clicks once to open the menu, and again to select the needed command regardless of how many menu levels exist.

Before you begin:

This task is optional.

1. Assess if menu commands should be regrouped or reorganized for better usability when rendered as mega menus. See [About the New User Interface \(UI\)](#) for more details about the mega menus.

You may need to modify the out-of-the-box My Desk menu and page-level menus. You may need to install megamenus first before evaluating the new display of the menus.

The megamenus only display icons at the top-level indent. Icons defined at any other level will be replaced with a bullet.

2. If an intermediate JSP is used, or a menu inside a category menu is used, pass the `categoryTreeName` URL parameter when opening the context page.

For example:

```
 ${COMMON_DIR}/emxTable.jsp?program=emxProduct:getDocuments
&table=SketchSummary&selection=multiple&sortColumnName=Name
&sortDirection=ascending&toolbar=ProductSketchesSummaryToolBar
&header=emxApparel.Header.ArtworkDetails&HelpMarker=emxhelpartworkdetails
&parentTypeName=type_Sketch,type_Graphic&FilterFramePage=
\${COMPONENT_DIR}/emxCommonDocumentCheckoutUtil.jsp&FilterFrameSize=1
&categoryTreeName=type_Products
```

3. Remove the `Row Select = multi` setting from menu command that many be disabled in the category tree or actions commands. Replace it by passing the `selection=multiple` URL parameter, as shown in this example:

```
{href=\${COMMON_DIR}/emxIndentedTable.jsp?inquiry=FAOCOLORSWhereUsed
&table=FAOCOLORWhereUsed&header=emxApparel.Header.ProductTemplateWhereUsed
&HelpMarker=emxhelpcolorwhereused&pagination=0&selection=multiple
&imgType=Thumbnail_small&chart=false&freezePane=MarketingName} \
```

Implementing Slide-in Dialogs and Edit Dialogs

The slide-in feature reduces the number of popup windows the user needs to deal with. The new UI encourages developers to use slide-in frames in place of pop-up windows where ever possible. In addition, you can convert popup dialogs used for editing to either slide-in frames, or edit-in-place windows (that is, the current page converts to edit mode instead of popping up or sliding in a new dialog).

Forms with multiple columns and custom JSP forms may not migrate correctly. They may be placed in a popup DIV that does not completely cover the calling page. As an alternative, you can migrate the form to a configurable component (with a single column) or migrate the form to an FS page component. See [Converting Pages to the New UI Framework](#).

1. In the Settings for the object that will open the window, enter this name/value pair:

Slide-in frames can be opened from a toolbar command or menu, navigation tree command or menu, table column, or form field.

Name: Target Location

Value: slidein

2. If designing custom pages and not using the configurable components, use the JavaScript API to target the slide-in frame and then to close that frame. Use the `top.showSlideInDialog(url, isModel)` API to open the slide-in frame; use `top.closeSlideInDialog()` for the Close/Cancel buttons to close the slide-in frame. See the JavaDocs for details.
3. When migrating a custom JSP to use the slide-in frame, you need to change the JavaScript that closes the dialog as shown in these before/after code snippets:

Before:

```
fs.createFooterLink("emxFramework.Command.Cancel", "parent.window.close()",  
roleList, false, true, "emxUIButtonCancel.gif", 0);
```

After:

```
fs.createFooterLink("emxFramework.Command.Cancel", "doCancel()",  
roleList, false, true, "emxUIButtonCancel.gif", 0);
```

```
function doCancel() {  
    top.closeSlideInDialog();  
}
```

4. When migrating an FS Page to use the slide-in frame, you need to update the targets for refreshing summary listing pages with the changes made by the user:

```
//Starts-for next Gen slidein  
String targetLocation = request.getParameter("targetLocation");  
targetLocation = (null == targetLocation ||  
"null".equals(targetLocation))?"":targetLocation;  
//Ends for next Gen slidein
```

```
String contentURL="emxEngrCreateBOMMarkupDialog.jsp?objectId=" + objectId  
+ "&mode=" + sMode + "&commandType=" + commandType +  
"&relId=" + relId + "&targetLocation=" + targetLocation;
```

In addition, you also need to make the following change to the content page that called the FS Page:

```
//Starts-for next Gen slidein  
String targetLocation = request.getParameter("targetLocation");  
targetLocation = (null == targetLocation ||  
"null".equals(targetLocation))?"":targetLocation;  
boolean slidein = targetLocation.equals("slidein")?true:false;  
//Ends for next Gen slidein
```

5. Evaluate popups used for editing a form:

- If the calling page is a form (such as a properties page), use edit-in-place instead of popping up a new dialog. You can use edit-in-place with forms that have multiple columns.

Note: Do NOT use a slide-in frame to edit a full page web form.
- If the calling page is a table or structure browser that includes edit buttons in the rows, use a slide-in frame to edit a selected object.
- If the calling page is a table or structure browser that includes an Edit command in a right-mouse button menu, use a slide-in frame to edit the selected object.
- If the calling page is a table or structure browser and the edit form contains multiple columns, keep it as a popup dialog.

6. To implement edit-in-place, define the `Target Location` setting for the Edit command.

Name: Target Location

Value: self

To implement slide-in edit frames, see the instructions in steps 1-4.

Removing Inline Style Definitions and Invalid HTML Elements

Inline style definitions and invalid HTML elements and element attributes may interfere with or override the formatting intended by the design of the application. Formatting or rendering errors may occur which result in overlapping graphic elements, text that overruns boundaries, misalignment of elements, and other errors. You can follow this task to prevent these issues.

Before you begin:

This procedure is optional.

1. Search for `<style>` and `style=` instances applied to an HTML element.

For example, search for entries such as these:

```
<style><!--  
body {  
    background-color: #DDDECB;  
}  
--></style  
or  
<td style="background-color: #DDDECB; font-weight: bold; color: #F00;">
```

2. Remove the inline styling and replace with references to styles in an external stylesheet. If necessary, new styles can be added to an externally referenced CSS file.

For example, this script in the `<head>` element of a custom page results in the screen shot:

```
<script language ="Javascript">  
addStyleSheet("emxUIDefault");  
addStyleSheet("emxUIList");  
</script>
```

The screenshot shows a web-based application interface titled "Bill of Material Change History ss:FA10". On the right side, there is a dropdown menu with several options: "Sort Order" (selected), "Component Location", "Raw Material", "Quantity", and "Net Usage". Below this menu, there is a date range selector with two dropdowns: "Last 30 Days" and "Specific Change". The "Specific Change" dropdown has a value "Clark : 2010-12-2 15:05:01". The main content area displays a table of change history. The table has columns for "Change Type", "Type", "Name", "Description", "Quantity Detail", "Article", "Supplier", "Location", "Usage", "Supplier Quantity", and "Supplier Size". There are four rows of data in the table, each representing a different type of fabric change.

Change Type	Type	Name	Description	Quantity Detail	Article	Supplier	Location	Usage	Supplier Quantity	Supplier Size
Bonded Fabric	BF-21-000	Bonded Fabric 0_1						-	1.0	0.0
Bonded Fabric	BF-21-001	Bonded Fabric-0_2						-	2.0	0.0
Bonded Fabric	BF-21-002	Bonded Fabric-0_3						-	3.0	0.0
Bonded Fabric	BF-21-003	Bonded Fabric-0_4						-	4.0	0.0

The changes required to fix the styles are:

- a. Add the CSS to apply to the table:

```
<table class="list">  
This replaces existing HTML such as <table border="0" width="100%">
```

If the page displays a form in view or edit mode, then the table element should use this: `<table class='form'>`

- b. Apply the needed CSS references:

```
<fw:mapListItr mapList="<% objMapList %>" mapName="objMap">  
<tr class='<framework:swap id = "1"/>'>  
// this is to display the alternative color for tablerows.
```

When the changes have been made, the screen renders correctly like this:

Bill of Material Change History ss:FA10

The screenshot shows a software interface titled "Bill of Material Change History ss:FA10". At the top right, there is a filter section with the following options: "Sort Order" (selected), "Component Location", "Raw Material", "Quantity", and "Net Usage". Below this is a date range selector with "Last 30 Days" selected and a specific change ID "Clark : 2010-12-2 15:05:01" entered. The main area displays a table of changes:

Change Type	Type	Name	Description	Quality Details	Product A	Supplier Location	Usage Supplier	Quantity	Size
Bonded Fabric	EF-21-0001	Bonded Fabric 0_1			-		1.0	0.0	
Bonded Fabric	EF-21-0001	Bonded Fabric 0_1			-		2.0	0.0	
Bonded Fabric	EF-21-0002	Bonded Fabric 0_2			-		3.0	0.0	
Bonded Fabric	EF-21-0003	Bonded Fabric 0_3			-		4.0	0.0	

3. Remove invalid HTML elements.

Although formatting can be accomplished using either HTML tags or Cascading Styling Sheets (CSS), the new UI almost exclusively relies on CSS for formatting and many formatting-oriented HTML tags are being deprecated for HTML 5. HTML formatting tags may work using the transitional DOCTYPE, however the new UI mandates a different DOCTYPE.

These tags should be removed:

- <address>
-
- <basefont>
- <big>
- <blink>
- <center>
-
-
- <hr>
- <i>
- <layer>
- <marquee>
- <s>
- <small>
- <strike>
-
- <style>
- <u>

4. Remove invalid HTML element attributes. These attributes should be removed:

- align
- border
- bottommargin
- cellpadding
- cellspacing
- height
- leftmargin
- marginheight
- marginwidth
- nowrap
- noshade
- rightmargin
- topmargin
- valign
- width

Note: In particular, review `<table>` and `` HTML elements for the `width`, `height`, and `border` attributes. Review `<td>` tags for the `width` and `border` attributes.

5. Remove non-breaking spaces () and spacer GIFs (utilSpacer.gif) used to space text or position page elements. Refer to the defined stylesheets for spacing and positioning styles.
6. Remove the size definitions of popup windows and define the `Popup Size` setting for the command that opens the popup

dialog.

For example:

Name: Popup Size

Value: MediumTall

This table lists the allowed values and the size (in pixels) of the popup dialog.

Accepted Value for the Popup Size Setting	Pixels (height x width)
Large	960x700
Medium	812x500
Small	512X500
SmallTall	412X700
MediumTall	812x700

Enabling Automatic Type Ahead

This features allows users to start typing in a field (instead of clicking a chooser button), and the type-ahead feature displays matching values in a drop-down list. A pop-up DIV tag displays the values, allowing the user to select a value once they have typed in enough characters to narrow down the result set to a manageable number. A chooser should still be available for times when the user may not know the needed value.

Before you begin: This task is optional.

1. Enable type ahead for form fields in any place where the user is required to enter a Person or Organization.

This example shows the RangeHrefs used to specify the pre-defined choosers:

```
 ${COMMON_DIR}/emxFullSearch.jsp?type=PERSON_CHOOSER&showInitialResults=false&selection=single  
 ${COMMON_DIR}/emxFullSearch.jsp?type=ORGANIZATION_CHOOSER&showInitialResults=false&selection=single
```

If an application uses a variation of the above chooser, ENOVIA recommends that you replace them with the standard chooser when possible. See [Automatic Type Ahead](#) for more details on implementing this feature.

2. A Business Administrator configures a property in emxSystem.properties to define the number of letters the user must type before values are retrieved. See the *Live Collaboration Administrator's Guide* for details on the type ahead system properties.

Updating PowerView Channel Heights

The height of PowerView channels is now specified as a percentage value. All existing PowerView pages need to be updated so that the channel heights are defined as percentage values instead of pixel values.

When defining a channel for a PowerView (emxPortal.jsp), prior releases required the height to be specified as a pixel value, with a default value of 260 pixels used if no height was specified. With this release, the Height parameter for a channel is specified as a percentage value, and the total of all stacked channels should not exceed 100.

If an existing PowerView page, with channels that have heights specified in pixels, is not updated to use percentage values, then the page will equally distribute the available space among the channels. In addition, if the total height of a set of stacked channels does not equal 100, then the page also equally distributes the available space among the channels.

You cannot specify the width of the channels -- width is distributed evenly among the channels.

Table-to-Structure Browser Conversion

The new user interface can automatically convert pages defined by emxTable.jsp to emxIndentedTable.jsp.

The emxSystem.properties file contains this property:

```
emxFramework.Table.Type=new
```

This property can be set to `new` or `classic`. When set to new (the default), Business Process Services automatically converts any page defined using emxTable.jsp to use emxIndentedTable.jsp. When set to classic, this conversion does not take place.

This property is set to `new`, it can be overridden for a single page by passing the `tableType=classic` URL parameter to emxTable.jsp.

If a table page uses a program or inquiry object, then the flat-mode structure browser is used for that table. If the table uses edit mode, emxTableEdit.jsp, then the structure browser in edit mode is used. If multiple programs are passed, then a program filter is added to the structure browser in the same way it had been added to the table as long as the tableMenu and expandProgramMenu URL parameters are also passed.

For reference, these features supported by tables are NOT supported by the structure browser component:

- pagination URL parameter, and all parameters associated with pagination, such as rememberSelection
- Filters that use FilterFramePage=emxAEFFilter.jsp. This functionality can be implemented using custom filters in toolbars.
- CancelButton URL parameter to close the browser window (for a popup window). Replace this parameter with the cancelLabel URL parameter.
- disableSorting URL parameter. Use sortColumnName=none.
- stopOOTBRefresh URL parameter
- FilterFramePage URL parameter
- FilterFrameSize URL parameter
- Plain HTML code for table headings. If used, the heading shows as plain text (displaying the HTML code). The HTML should be converted to XHTML for proper rendering.
- ByTable column settings NOT supported:
 - Group By
 - Column Type=checkbox. Use selection=multiple URL parameter if you need to include a checkbox for rows.

Extending Applications

This section presents some general coding guidelines to follow when working with JPOs, JSPs, and JavaBeans. It also explains how to extend these layers of logic.

In this section:

- [!\[\]\(d7dc477cd0116df0962645e742046120_img.jpg\) Java Coding Hints](#)
- [!\[\]\(988c6500c92117995800592d51f570fa_img.jpg\) Using JPOs](#)
- [!\[\]\(988449840be6c46c3508aadcbc0e0b94_img.jpg\) Round-trip Engineering](#)
- [!\[\]\(56710d20e1c3cbfb626dc9ca7271c668_img.jpg\) Using Servlets](#)
- [!\[\]\(572266eb68cf82c5cdc2efd40538675a_img.jpg\) JavaServer Pages](#)
- [!\[\]\(3f580f5bc969951e5e12681950e5a1f7_img.jpg\) About Mapping Files and JavaBeans](#)
- [!\[\]\(e97363bcb939fd3f028b19f4ecd5aa37_img.jpg\) Multiple Jar File Architecture](#)
- [!\[\]\(85e879714a0af8081dfd2dcd739b2deb_img.jpg\) Business Process Services Programming](#)
- [!\[\]\(b326f73944f2bb64ec0097f74b7c99eb_img.jpg\) Naming Conventions](#)
- [!\[\]\(f93e86d993b3430b230a2d458ae4fcc4_img.jpg\) Extending JPOs](#)
- [!\[\]\(fbee77ea2a7a35e283ace72897add317_img.jpg\) About Writing JSPs](#)

Java Coding Hints

This topic provides tips for working with Java to extend ENOVIA applications or create your own custom applications.

When customizing an ENOVIA product, you must keep an eye toward extending each layer of logic in such a way as to preserve the customizations across subsequent releases. You should customize Java code at the lowest possible level (JPO first, then Beans, then JSP as a last resort). Preserving custom code is accomplished by following these guidelines:

- JSPs: copying pages and using CustomFilter
- Beans: adding a derived class, updating a mapping file registration, leveraging polymorphism
- JPOs: placing code in derived JPOs

When business logic is not included in the JSPs, it is easier to incorporate a Front Controller architecture (like Struts). Other reasons to keep logic out of JSPs include performance, reuse, maintenance, and so on. There is a wealth of functionality contained in the Beans and base JPOs available for use by your custom code. Using the beans and JPOs rather than JSPs for business logic speeds your development and guarantees efficient use of the Studio Customization Toolkit.

You can classify JPO methods as either "invoked" or "called". Invoked methods are accessed through the Studio Customization Toolkit (using JPO.invoke()) or through MQL (execute command) and must adhere to two specific signatures (each take a Context and String[], one returns an int, the other an Object). For all other JPO methods, use the term "called" (or "non-invoked") and these methods should have normal signatures consisting of individual arguments. The invoked methods that return an Object can return anything derived from the Object which means they can return anything (except void).

In general, Java coding should be limited to writing JPOs and taking advantage of Bean methods.

Other tips:

- Group logic into classes, by type or functional area (some split UI out into a separate class)
- Methods should always take a Context.
- Avoid the use of deprecated methods, classes, and interfaces. Change existing code as time allows.
- Never invoke a JPO method from a JPO method (just call the other JPO method directly).
- JPOs should never call a Bean method that invokes a JPO method.
- Avoid using the DomainConstants interface.
- Follow the JPO Invoke (Facade) design pattern.
- Take advantage of the class concept in Java. Do not write a separate JPO for each existing Tcl program object (and rely on mxMain).
- Define symbolic names as a constant and always get the actual name in your code using getSchemaProperty(context, symbolic).

The actual names of schema objects are stored as constants in the DomainConstants interface (which DomainObject implements) so that they could be used in building select constants. This design allows a user to remap a symbolic name at any time, and the only way to pick up the change in static finals is to restart the server.

The naming convention for entries in DomainSymbolicConstants will be `SYMBOLIC_xxx` where xxx is the exact string the constant represents. For example:

```
public static final String SYMBOLIC_type_DocumentSheet="type_DocumentSheet";
```

- Deploy in development/expanded mode.
- Use an IDE like NetBeans, Eclipse, or WSAD.
- Use the JPO demangler tool (available through Solutions Library) to prepare extracted code for javadoc.
- Use jpotool to migrate code from a given class to a JPO.

Using JPOs

This section gives you some practical advice on using JPOs. Java Program Objects (JPOs) are written using the Studio Customization Toolkits for use with both the desktop and browser ENOVIA products.

In this section:

- [About JPOs](#)
- [Simple JPO Example](#)
- [Accessing the Studio Customization Toolkit](#)
- [Passing Objects into a JPO Method](#)

About JPOs

JPOs and the administrative objects that execute them are the basis for the ENOVIA Live Collaboration end-user internet applications. Web-centric administrative objects are also part of ENOVIA Live Collaboration. Administrators can create commands, menus, tables, web forms, and portals (as well as inquiries and channels, as supporting objects), to design a web-based implementation.

See [About Java Program Objects \(JPOs\)](#) for background details.

The following topics are discussed:

- [Runtime Program Environment](#)
- [TCL Wrappers](#)

Runtime Program Environment

The Runtime Program Environment (RPE) is accessible to JPO methods using the Studio Customization Toolkit MQLCommand object.

The various get and set environment commands can be passed and the results received back. The RPE is not dependable in the thin client architecture of ENOVIA products. The RPE should be avoided in this environment.



TCL Wrappers

You need to write individual Tcl Program objects for each method of a Business Type.

However, these individual Tcl Program objects should just be wrappers that call the appropriate JPO method. Likewise, Tcl Program wrappers should be used on the command bar to call JPO methods.

These Tcl wrappers can use macros that are then passed to the JPO method as arguments. This is necessary since JPOs are compiled and therefore lose any use of macro substitution. The Tcl wrapper gets the appropriate macro values at runtime and passes them along as arguments to the JPO method.

Simple JPO Example

This section shows a simple JPO example.

Run Business Modeler as a Business Administrator and place the following code into a new Program Object named "Hello World". Mark the Program Object as type "Java."

```
import matrix.db.*;
import java.io.*;
public class ${CLASSNAME}
{
    public ${CLASSNAME} () {
    }
    public ${CLASSNAME} (Context context, String[] args) throws
Exception
    {
    }
    public int mxMain(Context context, String []args) throws
Exception
    {
        BufferedWriter writer = new BufferedWriter(new
MatrixWriter(context));
        writer.write("Hello World!\n");
        writer.flush();
        return 0;
    }
}
```

There are several ways to invoke this new JPO. In this example, an instance of the class "Hello World" is instantiated using the constructor (Context, string[]). Actually, the resulting class name will be modified to adhere to the Java language rules for naming classes. This process is referred to as *name mangling*.

The Context object is constructed by the kernel to allow the program to share the transactional context of the launching thread. The constructor can be implemented to connect to a Live Collaboration kernel elsewhere on the network. In this case, however, the program will run on a different thread, and inside its own transaction boundary.

Accessing the Studio Customization Toolkit

You can access JPOs through the Studio Customization Toolkit through two invoke methods on a new JPO class.

One form of the invoke method returns an int (useful for returning exit codes).

```
int ret = JPO.invoke(Context context, String className, String[]  
initargs,  
String methodName, String[] methodargs);
```

The other form returns a Java Object.

```
Object ret = JPO.invoke(Context context, String className,  
String[] initargs,  
String methodName, String[] methodargs,  
java.lang.Class retType);
```

An example of an object method in the emxProject JPO would be:

```
import matrix.db.*;  
public class ${CLASSNAME} {  
    public ${CLASSNAME} (Context context, String []args) {  
    }  
    public BusinessObjectList query(Context context, String []args)  
    {  
        Query query = new Query(args[0]);  
        return query.evaluate(context);  
    }  
}
```

called by the following in a .jsp:

```
String[] init = new String[] {};  
String[] args = new String[] { ".finder" };  
BusinessObjectList list =  
(BusinessObjectList)JPO.invoke(context, "Project",  
    init, "query", args, BusinessObjectList.class);
```

Use casting because the system only knows to return an object instance. In the example above, the BusinessObjectList is consistently used for declaring the object to hold the return object, for casting, and for the sixth argument that defines the return type.

Passing Objects into a JPO Method

The JPO class has two methods that correspond to serializing an Object into a String, and de-serializing a String back into an Object.

The method names are `packArgs` and `unpackArgs`, respectively. Since JPOs only accept strings as arguments, you must use the `packArgs/unpackArgs` methods to convert Java objects to/from strings when using JPOs.

These methods actually work with a String array of size two, that holds the class in the first slot and the serialized object in the second slot. This allows users of the String array to do proper type casting. One can build up several Objects in a String array by calling `packArgs()` several times using the proper indexing into the String array. However, it is better to place all arguments in a single compound object (like a hashmap) and then pack just the single compound object into the args array. Keep in mind that any Object needing to be passed as an argument in this way must implement the `java.io.Serializable` interface.

You can only call `packargs` on objects that are serializable. Most classes are now serializable and so may be passed as arguments to JPOs.

The `packArgs` method has the following signature:

```
public static String[] packArgs(Object in) throws Exception
```

The `unpackArgs` method has the following signature:

```
public static Object unpackArgs(String[] in) throws Exception
```

The following packing logic is placed in a Bean that is called from a JSP.

```
/**  
 * Sends an icon mail message to the specified users.  
 *  
 * @param context the Matrix <code>Context</code> object  
 * @param toList the to list of users to notify  
 * @param ccList the cc list of users to notify  
 * @param bccList the bcc list of users to notify  
 * @param subject the notification subject  
 * @param message the notification message  
 * @param objectIdList the list of objects to send with the  
 notification  
 * @throws FrameworkException if the operation fails  
 * @since AEF 9.5.0.0  
 * @grade 0  
 */  
public static void sendMessage(Context context,  
                           StringList toList,  
                           StringList ccList,  
                           StringList bccList,  
                           String subject,  
                           String message,  
                           StringList objectIdList)  
throws FrameworkException  
{  
    try  
    {  
        ContextUtil.pushContext(context);  
        // Create the arguments for the notification.  
        Map note = new HashMap();  
        note.put("toList", toList);  
        note.put("ccList", ccList);  
        note.put("bccList", bccList);  
        note.put("subject", subject);  
        note.put("message", message);  
        note.put("objectIdList", objectIdList);  
        // Pack arguments into string array.  
        String[] args = JPO.packArgs(note);  
        // Call the jpo to send the message.  
        JPO.invoke(context, "emxMailUtil", null,  
"sendMessage", args);  
    }  
    catch (Exception e)
```

```

    {
        throw (new FrameworkException(e));
    }
    finally
    {
        ContextUtil.popContext(context);
    }
}

```

The unpacking logic is then found in the JPO method where it is performed prior to passing the arguments on to the worker method (this design pattern is very useful for hiding the details of packing and unpacking arguments).

```

 /**
 * Sends an icon mail notification to the specified users.
 *
 * @param context the Matrix <code>Context</code> object
 * @param args contains a Map with the following entries:
 *           toList - the list of users to notify
 *           ccList - the list of users to cc
 *           bccList - the list of users to bcc
 *           subject - the notification subject
 *           message - the notification message
 *           objectIdList - the ids of objects to send with the
notification
 * @returns nothing
 * @throws Exception if the operation fails
 * @since AEF 9.5.0.0
 */
public static int sendMessage(Context context, String[]
args)
    throws Exception
{
    if (args == null || args.length < 1)
    {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    sendMessage(context,
        (StringList) map.get("toList"),
        (StringList) map.get("ccList"),
        (StringList) map.get("bccList"),
        (String) map.get("subject"),
        (String) map.get("message"),
        (StringList) map.get("objectIdList"));
    return 0;
}

```

Round-trip Engineering

The goal is to support true round-trip engineering, meaning that JPO code could freely move in and out of the ENOVIA Live Collaboration from an external IDE. The benefits of such an environment are obvious and include better editing, cross-referencing, visual modeling, and debugging.

In this section:

- ❑ [Compiling for Round-trip Engineering](#)
- ❑ [Editing to Support Round-trips](#)
- ❑ [Extracting to Support Round-trips](#)
- ❑ [Inserting to Support Round-Trips](#)
- ❑ [Inserting and Extracting to Support Round-Trips](#)
- ❑ [Debugging JPOs for Round-trip Engineering](#)
- ❑ [Java and exec Commands](#)

Compiling for Round-trip Engineering

ENOVIA Live Collaboration is set up so that invoking JPOs will cause an automatic compile if necessary, using Java directly.

Two ENOVIA Live Collaboration ini variables are related to compiling JPOs:

- MX_JAVAC_FLAGS allows you to pass extra flags to the compiler.
- MX_JAVA_DEBUG, when set to `true`, compiles JPOs with debugging information available and sets up MQL so that a debugger program can be attached to it.

When debugging an external JPO, you must also set `matrix.jpo.external` to `true` when passing to the JDK that launches your application server and add the path to the compiled mxJPO classes to `MX_CLASSPATH`.

The `matrix.jpo.external` setting allows you to execute JPOs outside your database. When set to `true`, the classloader will look on disk (in `MX_CLASSPATH`) before looking in the database to find a JPO's byte code.

To force a compile before invoking a JPO method, use the MQL command `compile program`. This is useful for bulk compiling and testing for compile errors after an iterative change to the JPO source.

```
compile program PATTERN [force] [update] [COMPILER_FLAGS];
```

When a JPO is compiled or executed, any other JPOs which are called by that JPO or call that JPO must be available in their most recent version. The `compile` command includes an `update` option that updates the requested JPO's dependencies on other JPOs that may have been added, deleted, or modified. Use the following command to ensure all JPOs have up-to-date dependencies:

```
compile prog * force update;
```

The class file resulting from the `compile` command is placed into the ENOVIA Live Collaboration database. A special ENOVIA Live Collaboration class loader finds the class file in the database when referenced.

When a password trigger uses a JPO, the program must be compiled explicitly before it can be used, since there is no active context before a login transaction is committed. Business Administrators must compile the program when initially written and whenever modifications are made. Include the `force` option when you need to recompile a JPO. In order to recompile the JPO(s) used by password trigger, the Business Administrator can set `MX_PASSWORD_TRIGGER = false` in `envia.ini` for the Studio Modeling Platform and log in through MQL.

As of Version 10, `MX_JAVAC_OPTIONS` is set to `-nowarn`, which suppresses deprecation warnings when compiling JPOs.

If you want to see deprecation warnings, for example, when debugging:

- For automatic compiles, you can set `MX_JAVAC_OPTIONS` to `-deprecation`.
- For compiles using the command line, include the `-deprecation` flag and precede the `compile` command with `verbose on`, for example:

```
verbose on  
compile prog NAME -deprecation
```

Editing to Support Round-trips

From Business Modeler, an external editor of choice can be launched from the Program object dialog. The enovia.ini variable MX_EDITOR is used to label the button, and the variable MX_EDITOR_PATH is used to specify the path to the external editor. Once specified, the dialog includes an additional button used to launch the editor.

All changes made to the code are automatically read back into ENOVIA Live Collaboration. As a simple example on the Windows operating system, the settings might be:

```
MX_EDITOR=notepad  
MX_EDITOR_PATH=c:\winnt\system32\notepad.exe
```

Another approach is to extract the Java code out of the ENOVIA Live Collaboration database and into a file directory. This can be done using the [extract program](#) MQL command. The resulting code will be ready for compilation (name mangling will already have been performed) and further development.

You may choose to do all, or most of your editing in an offline IDE. The only requirement is that JPO names must follow the Java language naming convention. While editing outside ENOVIA Live Collaboration, the names of JPO classes can be referenced in a semi-mangled form. This form consists of the JPO name followed by the string "_mxJPO". For example, if editing code that needs to instantiate an object defined by the JPO "emxProject", the following would be used:

```
emxProject_mxJPO proj = new emxProject_mxJPO();
```

JPOs in ENOVIA Live Collaboration can be populated with external files using the [insert program](#) MQL command. This command converts mangled names back into the appropriate special macro form. The previous example would be changed to:

```
 ${CLASS:emxProject} proj = new ${CLASS:emxProject}();
```

Extracting to Support Round-trips

Extracting the Java source in the form of a file out to a directory is useful for working in an IDE.

While in the IDE a user can edit, compile, and debug code. The `extract program` MQL command processes any special macros and generates a file containing the Java source of the JPO. If no source directory is given, the system uses ENOVIA_INSTALL/java/custom (which is added to `MX_CLASSPATH` automatically by the install program).

```
extract program PATTERN [source DIRECTORY] [demangle]
```

To use the extract feature, the JPO name must follow the Java language naming convention (that is, no spaces, special characters, and so on). Only alphanumeric printable characters and '.' and '_' are allowed in class names in the JPO.

The demangle option causes the file and class names to be normalized during the extract. For example, an extract of the HelloWorld JPO by default would create a file named `HelloWorld_mxJPOTwEAAAFAAAA.java` containing a class named `HelloWorld_mxJPOTwEAAAFAAAA`. When using the demangle option, the extracted file will be named `HelloWorld_mxJPO.java` containing a class named `HelloWorld_mxJPO`. This ability is needed by some IDEs that do not deal well with the hash value added to the file and class names.

Program object names may include a package specification, such as `matrix.java.custom.testjpo`. In this example, `matrix.java.custom` is the package, and `testjpo` is the name of the class. In this case, the code of the JPO does not need to include a package statement; it is added to the code if necessary during compilation or extraction. In addition, when programs are extracted, the source is placed in an appropriate subdirectory based on the name of the JPO; for example, ENOVIA_INSTALL/java/custom/testjpo.java.

Inserting to Support Round-Trips

After testing and modifying Java source in an IDE, you need to insert the code back into JPOs in the ENOVIA Live Collaboration database.

The `insert program` MQL command regenerates special macros in the Java source as it is placed back into a JPO (reverse name-mangling; see [Macros](#)). If the JPO does not exist, the `insert` command creates it automatically.

```
insert program FILENAME | DIRECTORY;
```

For example:

```
insert program ENOVIA_INSTALL/java/custom/testjpo_mxJPO.java
```

or

```
insert program ENOVIA_INSTALL/java/custom/
```

The latter will insert all the .java files in the specified directory.

When programs are inserted, the package statement is stripped from the source code, and the package name is prepended to the name of the class to create the program object; in the above example, it will be named `ENOVIA_INSTALL.java.custom.testjpo`.

Note: To use the insert feature, the JPO source must follow the Java language naming convention for class code (that is, no spaces, special characters, and so on) and also include "`_mxJPO`". All characters including and after "`_mxJPO`" are removed during insertion (refer to [Macros](#).)

Inserting and Extracting to Support Round-Trips

Macro processing is used by all program objects (JPO as well as Tcl) before they are run to allow value replacement for a macro in a program.

For a broader description of Macro Processing, refer to [Macros](#). This means that if a JPO is written with a command that has 3 backslashes, when it is executed within ENOVIA Live Collaboration it goes to Java with 2 backslashes.

The extract command calls the macro processing code, so the files it produces can be executed by Java outside of ENOVIA Live Collaboration. This means that the insert command also needs to take macro processing into account. Prior to Version 10.7, the insert command doubled up any backslashes in JPO code going into the database and halved the number when using the extract command. When pre-version 10.7 databases are upgraded to version 10.7 or higher, all JPOs are tagged internally as "old". When "old" JPOs are extracted, the backslashes are halved, but when inserted back into Version 10.7 or higher the backslashes are no longer doubled, and the JPO is tagged "new." Macro processing of new JPOs (that is, programs tagged "new", or those newly inserted) with Version 10.7 or higher is all done internally.

Important: If backslashes are not appropriately manipulated for your programs, check the number of backslashes used in the code and adjust as needed.

Debugging JPOs for Round-trip Engineering

JPOs can be debugged using an external IDE.

As stated earlier, you must set the .ini variable MX_JAVA_DEBUG to True, and then run the MQL console and force compile the JPOs to be debugged. You can then attach to the JVM, started on behalf of the MQL console, on port 11111 from any number of IDEs (such as NetBeans or Eclipse).

To debug in an application server environment, you must determine the technique specific to each type of application server on how to setup the JVM for debugging. You must run in the RIP mode version of RMI so that the JPOs run in the same JVM as the application server. This makes it easier to debug your Java business logic that might be spread across JSPs, Beans, and JPOs, since only 1 JVM is running.

When debugging an external JPO, you must also set matrix.jpo.external to true when passing to the JDK that launches your application server and add the path to the compiled mxJPO classes to MX_CLASPATH.

Java and exec Commands

You can use the `java` command and the `exec prog` command for JPOs.

The syntax for these commands is as follows:

```
java CLASS_NAME [-method METHOD_NAME] [ARGS] [-construct ARG];  
  
exec program PROGRAM_NAME [-method METHOD_NAME] [ARGS] [-construct ARG];
```

where `ARGS` is zero or more space-delimited strings and `ARG` is a single string. If the method returns an int, then this is assumed to be the exit code from the method (which has meaning to the ENOVIA Live Collaboration when used as a trigger program, wizard utility program, etc.). If the method returns an Object, then the exit code is set to zero and the object is serialized and passed back in the program's output stream. This serialized object is of no use in the MQL environment, but is very useful in the Studio Customization Toolkit environment (see next section).

The `java` command takes any class name and finds that class both inside a JPO and outside the ENOVIA Live Collaboration database. The class name must be in its fully mangled form.

The `classname selectable` can be used on JPO programs to print the classname specified.

The `exec program` command can be used to avoid having to know the mangled class name. However, the class has to live inside a JPO (no search is made outside the ENOVIA Live Collaboration database).

The `-construct` clause is used to pass arguments to the constructor. If more than one argument needs to be passed to the constructor, the `-construct` clause can be repeated as many times as necessary on a single command. The `construct` clause may be necessary to establish an object's identity. For example, if a JPO named "emxProject" held business logic for managing business objects of type "Project", then this JPO would undoubtedly be derived from the AEF `emxDomainObject` JPO (which derives from the Studio Customization Toolkit `BusinessObject` class).

```
import matrix.db.*;  
public class ${CLASSNAME} extends ${CLASS:emxDomainObject} {  
    public ${CLASSNAME}(Context context, String[] args) throws  
        Exception  
    {  
        super(args[0]);  
    }  
    public int ComputeEndDate(Context context, String[] args)  
        throws Exception  
    {  
        // do work here to compute this Project's end date?  
        return 0;  
    }  
}
```

With this assumption, to compute the end date of an existing Project, the following command would be executed inside a Tcl method on the schema type Project:

```
exec prog Project -method ComputeEndDate -construct ${OBJECTID}
```

You can launch methods on a business object so that the term "method" is not overloaded when using a JPO method. This syntax allows either the keyword "method" or "class" to be used. For a JPO , you should use the keyword "class":

```
exec bus T N R class Project -method ComputeEndDate -construct  
${OBJECTID}
```

Once constructed, an object stays in existence until the transaction is completed. Subsequent method calls do not need to supply the `-construct` clause.

Using Servlets

Similar to the way Java applets extend the functions of a browser, Java servlets extend the functions of Web/application servers. The ENOVIA Live Collaboration Servlets are installed as part of the ENOVIA Live Collaboration Server (eMatrixServletRMI) and directly call the Studio Customization Toolkit, as is the case with the other parts of the server. Servlets must be registered with the Web/application server.

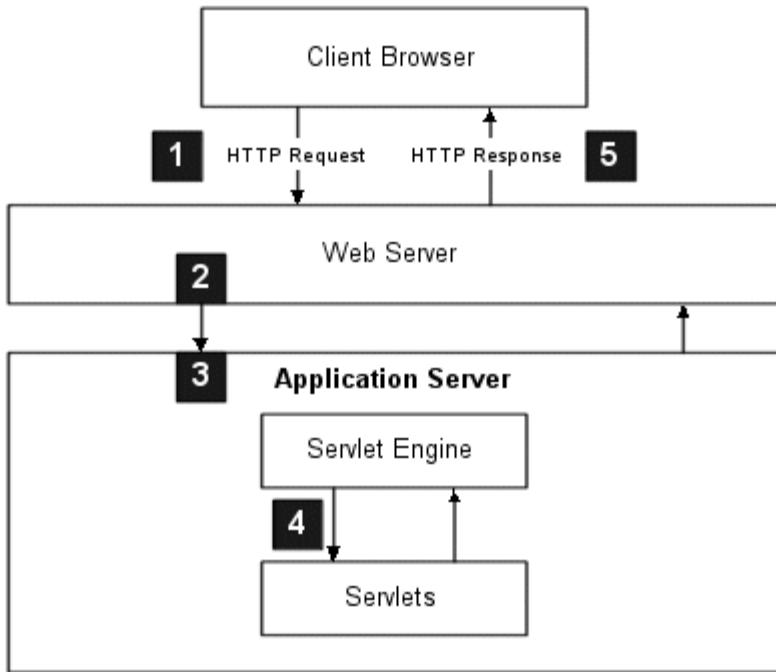
In this section:

- [Servlet Processing](#)
- [Accessing the Servlet](#)

Servlet Processing

This section presents an overview of servlet processing.

The following graphic illustrates the processing that occurs when a servlet is requested from a browser. Each numbered point is described below. (The servlet engine could be part of the Web server and not use an application server).



1. A browser requests the URL (alias) for a servlet, such as `http://HOSTNAME:PORT/enovia/servlet/login`. This request could come from within a JSP, another servlet, an HTML page, or directly from the URL.
2. The Web server (WebLogic, IBM HTTP, Apache, IIS) recognizes the servlet as a program that should be passed to the application server because servlets are registered with the application and Web server.
3. The Web server forwards the request to the application server (WebLogic, WebSphere, etc.), which manages the servlet engine.
4. The servlet engine converts the browser request into a `HttpServletRequest` Java object, which is passed to the servlet along with an `HttpServletResponse` object that is used to send the response back to the browser.
5. The servlet then generates the response to the browser, which is typically HTML and usually includes dynamic content generated by the servlet processing.

Accessing the Servlet

The sample servlet is saved as a Java file, `HelloeMatrixServlet.java`. To run the servlet, follow the steps in this task.

1. Compile the servlet, as you would any Java file, to produce a *.class file. Make sure the packages that are imported in the servlet, javax.servlet and javax.servlet.http, are included in the CLASSPATH environment variable. If you do not have the classes, install them separately by downloading the JSRK (Java Servlet Development Kit), <http://java.sun.com/products/servlet/index.jsp>.

For example, in Windows, compile the servlet by entering the following command at the command prompt:

```
javac HelloeMatrixServlet.java
```

The result is a `HelloeMatrixServlet.class` file.

2. Copy the class file to the appropriate location within your Web or application server directory structure. For example, `c:\weblogic\myserver\servletclasses`.
3. From your Web browser, access the servlet by requesting the URL for the servlet. For example, `http://SERVER_HOST_NAME:PORTNUMBER/servlet/HelloeMatrixServlet`.

JavaServer Pages

JavaServer Pages (JSP) provide a simple programming method to display and dynamically update content on a Web page.

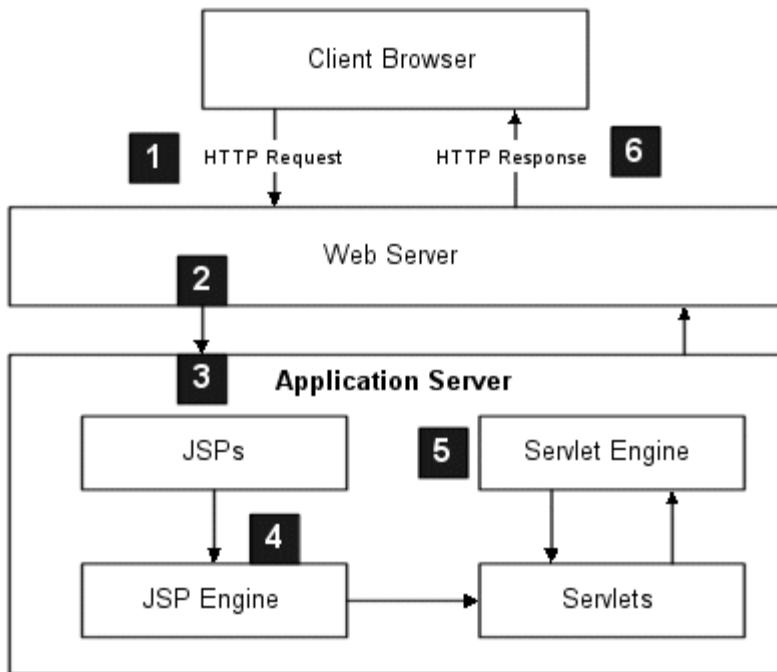
See [About JSPs](#) for basic details. The following topics are discussed:

- [JSP Processing](#)
- [JSP Access](#)
- [Custom JSP Filter](#)
- [Storing Custom Data](#)

JSP Processing

This section illustrates the processing that occurs when a JSP is requested from a browser.

Each numbered point is described below.



1. A browser requests the URL for a JSP, such as `http://HOSTNAME:PORT/enovia/eServiceLogin.jsp`.
2. The Web server (WebLogic, IBM HTTP, Apache, IIS) recognizes the `.jsp` extension in the URL, identifying the request as a JavaServer Page which must be handled by the JSP engine.
3. The Web server forwards the request to the application server (Jrun, WebLogic, WebSphere), which manages the JSP translator and servlet engine.
4. The page is translated into a Java source file and then compiled into a servlet class file. The compilation and translation phase only occurs when the JSP is first called (and the server finds no corresponding class files for the JSP) or when the source JSP changes.
5. The servlet engine converts the browser request into a `HttpServletRequest` Java object, which is passed to the servlet along with an `HttpServletResponse` object that is used to send the response back to the browser.
6. The servlet then generates the HTML-only response to the browser, including any static HTML included in the JSP and any dynamic content generated by the servlet processing. If you view the source of a JSP from your browser, you will see only HTML content. All Java and Javascript has been converted to static HTML.

This sample JSP demonstrates this process:

```
1 <%-- HelloENOVIA.jsp -- displays "Hello ENOVIA Client Application!"--%>
2
3 <HTML>
4   <HEAD>
5     <TITLE>Hello ENOVIA!</TITLE>
6   </HEAD>
7   <BODY>
```

```

8   <h1>Hello ENOVIA Client Application!</h1>
9   </BODY>
10  </HTML>
11  <%@include file = "ServletInfo.html"%>

```

Lines 1 and 2 are hidden JSP comments. These comments are ignored by the compiler and therefore do not appear in the HTML source in the browser.

Lines 3 through 10 contain standard HTML to produce the title and main text.

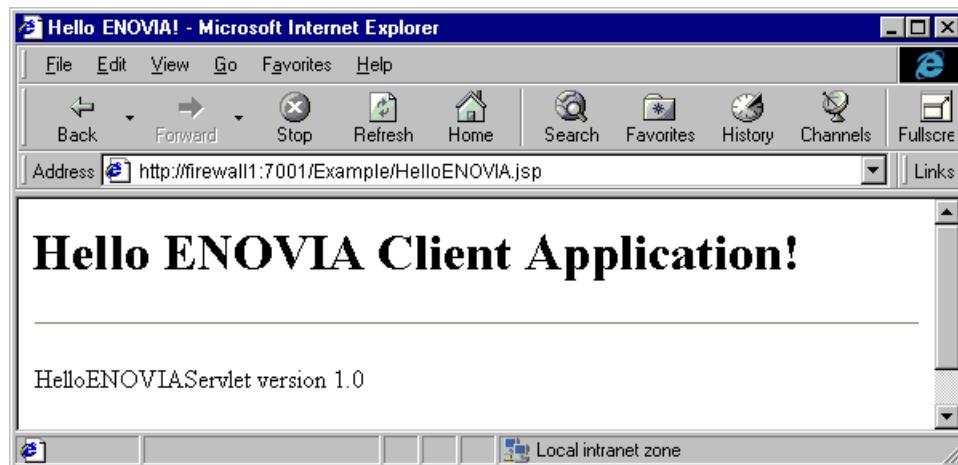
Line 11 uses an include directive to include the ServletInfo.html page. The ServletInfo.html contains only two lines to make up the same footer information displayed in the sample servlet:

```

<hr>
<p>HelloENOVIAServlet version 1.0</p>

```

The output for the JSP looks the same as the servlet output.



The HTML source for the JSP contains only HTML code. All Java and Javascript is replaced with static HTML. Also notice the hidden comments are not included.

```

<HTML>
  <HEAD>
    <TITLE>Hello ENOVIA!</TITLE>
  </HEAD>
  <BODY>
    <h1>Hello ENOVIA Client Application!</h1>
  </BODY>
</HTML>
<hr>
<p>HelloENOVIAServlet version 1.0</p>

```

Another way to produce the same output using JSP is to call the Hello ENOVIA servlet from a JSP:

```

<%-- HelloENOVIA1.jsp -- displays "Hello ENOVIA Client
Application" by calling the Hello ENOVIA servlet--%>
<html>
<!-- Forward to a servlet using jsp forward statement-->
<jsp:forward page="/servlet/HelloENOVIAServlet" />
</html>

```

JSP Access

This section describes how to access a JSP.

To access the JSP file, place the file, including any associated resource files, within the doc root directory structure of your Web server. Then request the URL for the JSP. For example, if the HelloENOVIA.jsp file is located in a directory called "Example" in the doc root, the URL would be: WEB_SERVER_DOC_ROOT/Example/HelloENOVIA.jsp.

Custom JSP Filter

The ENOVIA Live Collaboration Server contains a filter class that allows requests from ENOVIA products to check for a prefixed file name for custom JSP pages.

You can store custom resources (JSP, HTML, and image files) with prefixed filenames to prevent them from being overwritten

on subsequent application installs. You can even define multiple prefixes in order to keep various customization, accelerator, or integration files easy to identify.

You can implement a custom directory to store custom files; however, when using custom directories, there are certain jsp directives that will result in run-time (versus compile time) errors if the relative pathing is not correct. Therefore, ENOVIA recommends that you use custom prefixes instead of a custom directory.

Implementing the custom JSP filter involves simple changes to the web.xml file for your application server's WAR file. Refer to the *ENOVIA Live Collaboration Installation Guide* for details.



Storing Custom Data

You can store custom data or key/value pairs on the Java Context object in an application tier JSP or executable.

This custom data is then available on the server and in JPOs. The Context object (matrix.db.Context) allows programs to share the transactional context of the launching thread. For example, you could store custom data that includes a user's login site location on a JSP and have it available in JPOs.

Custom data stored in a frame context is not available from the parent context associated with that frame context via the application server session id. Custom data stored in a parent context is not available from any existing frame contexts associated with that parent context but will however, be available from any new frame context created under that parent context. Additionally, there are methods available under FrameworkServlet that enable JSPs to store custom data in the main context from which ALL frame contexts are derived.

Refer to the Javadocs for the classes and methods used in storing custom data.

About Mapping Files and JavaBeans

Use the newInstance method to serve instances of javabeans.

See [About JavaBeans](#) for background details.

Each jar file ships with a mapping file used by the newInstance method to return an instance of a Bean based on a given type or a given id. The newInstance method found in DomainObject is used for serving up instances of Beans derived from DomainObject based on a business type or an object id. The newInstance method found in DomainRelationship is used for serving up instances of Beans derived from DomainRelationship based on a relationship type or a rel id.

The mapping file naming convention uses an emx prefix and a MappingFile.properties suffix. In between is the name of the application or the word "Common". For example, the mapping file found in the engineering.jar file is named:

```
emxEngineeringMappingFile.properties
```

The mapping file found in the common.jar file is named:

```
emxCommonMappingFile.properties
```

The newInstance methods (found in both DomainObject and DomainRelationship) use these mapping files to serve up a new object instance of the appropriate class. The methods take an additional string argument that specifies the application name. This is used to find the application-specific version of a common components class. The search algorithm of the newInstance method guarantees the application-specific class is returned. The system first looks for the given type and then moves up the type hierarchy defined in the schema until a match is found. If no such class is found, the system continues to look for the given type in common components and then moves up the type hierarchy defined in the schema until a match is found. If no such class is found, the appropriate base class (DomainObject or DomainRelationship) is returned.

The Bean lookup method also includes a way for one or more Beans to be found before all other Beans when using the newInstance() method. This is a requirement for integrations to ensure that their Beans will always be found first. The mappings are included in an external file that can be managed independently of the ENOVIA products.

A property in emxSystem.properties lists the external mapping files that should be looked at first. The emxFramework.BeanMapping.PreliminarySearchFiles property can be used to list one or more mapping files that will be searched before falling back on the standard search algorithm, which looks at the mapping files associated with the Bean packages shipped with the ENOVIA products. The order in which the mapping files are given is important. If you want a mapping file searched first, then it must be listed first. In the example below, customMappingFile.properties will always be searched first. Then emxIntegrationsBeanMappingFile.properties will be searched. Finally, if no entry is found, the standard search will take place across the mapping files shipped with the applications. By default, the property is set to emxIntegrationsBeanMappingFile.properties.

```
emxFramework.BeanMapping.PreliminarySearchFiles =
customMappingFile.properties,emxIntegrationsBeanMappingFile.properties
```

This hard override, finding the appropriate Bean for a given type using newInstance(), blocks the standard search algorithm which begins in the mapping file of the given application and then moves on to the common mapping file to find a matching entry. It is important to understand that an entry placed in the mapping file associated with the emxFramework.BeanMapping.PreliminarySearchFiles property will override all Beans for a given type. For example, if the following mapping files all had an entry for type Part as shown below...

```
customMappingFile.properties ----->
type_Part= com.matrixone.apps.custom.CustomPart
emxIntegrationsBeanMappingFile.properties ----->
type_Part= com.matrixone.apps.integrations.IntegrationsPart
emxEngineeringMappingFile.properties ----->
type_Part= com.matrixone.apps.engineering.Part
emxCommonMappingFile.properties ----->
type_Part= com.matrixone.apps.common.Part
```

...then the newInstance() method would always return com.matrixone.apps.custom.CustomPart, even when called with a third argument set to engineering.

Multiple Jar File Architecture

This section describes working with the multiple jar file architecture.

The following topics are discussed:

- [UseBean Tag Changes](#)
- [New Up Classes](#)
- [Fully-qualified Class References](#)
- [Tag Library](#)

UseBean Tag Changes

This section describes the useBean tag and the newInstance method.

Use the `newInstance` method to instantiate Business types and Relationship types classes rather than the useBean tag when only page or request scope is required. This allows full use of the mapping files and type hierarchy lookup to obtain the appropriate object instance. The `newInstance` method cannot be used to instantiate classes from the ui, taglib, or util packages, or any helper classes that do not represent a Business or Relationship type.

The use of the `newInstance` method insulates your code from any reorganization of jar files that may happen. The `newInstance` method works off a mapping file that will be adjusted to point to the correct class.



New Up Classes

At no time should any Java code new up a Business or Relationship type object directly. The `newInstance` method must be used in these cases.



Fully-qualified Class References

There may be places in existing Java code where references are made to fully-qualified class names. Typically, these occur when referencing static methods. These areas should be modified to use the current packaging.

There may also be places in Java code where you directly new up instances of non-Business or Relationship type classes. These areas also should be modified to use the current packaging.



Tag Library

The tag library descriptor (tld) file named "component.tld." is referenced in all JSPs by a common include file that defines the tld source and available prefixes ("framework", "fw", and "emxUtil"). The fw and emxUtil prefixes have been removed, so you should use the framework prefix on any JSP extensions.

Business Process Services offers the common include file named emxTagLibInclude.inc that each application then includes in one of their common include files using:

```
<%@include file = ".../emxTagLibInclude.inc"%>
```

Business Process Services Programming

If you have any ENOVIA product, such as Supplier Central or Program Central, then you have Business Process Services. It is also possible to have Business Process Services only and build custom applications to work within it.

Business Process Services makes the job of creating custom JSPs much easier. It provides:

- a common schema for applications, including data models, business rules, and processes
- a standard user interface that includes menus and graphics
- mechanisms for:
 - controlling which menus and options are displayed for a user based on the user's assigned roles
 - looking up objects even if the object names have changed
 - managing multiple trigger events
 - automatically naming objects as they are created
- templates for JSP pages that perform common functions needed for all framework JSPs, such as presenting the login page if the user is not logged in, importing Java packages, and displaying the standard user interface
- utility trigger programs commonly needed, such as a program that enters the name of an object's originator in an Originator attribute and programs that check for required connections, files, or states
- wizards that let you add new suites, applications, and tasks, assign access to tasks, and merge existing schema with the framework schema
- sample data based on use cases (helpful for training and testing)
- installation program that prevents problems with object name collisions, tracks the version of objects so only updated objects are installed, and records details to a log file
- many ways to customize the UI to fit your business rules and applications

You can use this guide to help you write custom JSPs and servlets using the ENOVIA Live Collaboration servlet Studio Customization Toolkit. Keep in mind that Business Process Services supplies a common interface and uses specific conventions. To make sure your JSPs perform the common functions needed for every JSP, you should use the template JSPs installed with the Business Process Services. Also, to make a new application and its commands appear in the ENOVIA product menus, you need to register them and assign access to roles.

Naming Conventions

ENOVIA Live Collaboration classes use a standard naming convention to make it easier to find business logic and utility methods.

For business logic, there are three basic classes for each type found in the schema: 1 front-end Bean and 2 back-end JPOs (the base class and derived class). These 3-tuples make up the majority of the Beans and JPOs. The Bean name mimics the type name. The JPO names contain the Bean name, but also include a prefix of emx and often the package name. The base JPO name has a suffix of Base.

The remainder of the classes typically act as containers for utility methods. The utility methods are bundled by the function they were performing or the type of administration object they were acting on. The name of these classes contain the functional identification or name of the administration object, and are typically followed by the Util suffix. These Beans are found in the util subdirectory.

For example:

- Part Beans
 - com.matrixone.common.Part
 - com.matrixone.supplier.Part
 - com.matrixone.engineering.Part
- Part JPOs
 - emxCommonPartBase, emxCommonPart
 - emxSupplierPartBase, emxSupplierPart
 - emxPartBase, emxPart
- Part JPO trigger method:

```
* @trigger PolicyECPartStateReviewPromoteAction
public void floatEBOMToEnd(Context context, String[] args)
```
- Object in database:
 - T: eService Trigger Program Parameters
 - N: PolicyECPartStateReviewPromoteAction
 - R: floatEBOMToEnd

Extending JPOs

Java Program Objects (JPOs) have been designed to allow extensions to business logic. To preserve code from release to release, use class derivation.

In this section:

- [About Extending JPOs](#)
- [Extending Beans](#)
- [Extending JSPs](#)

About Extending JPOs

Java Program Objects (JPOs) have been designed to allow extensions to business logic. To preserve code from release to release, use class derivation.

The following topics are discussed:

- [Base Class and Derived Class Combination](#)
- [Extending a JPO Method](#)
- [JPO Hierarchy](#)

Base Class and Derived Class Combination

JPOs ship as a combination of a base class and a derived class.

For example, the JPO responsible for Part related business logic is delivered as the combination of a base class named emxPartBase and a derived class named emxPart. The JPO naming convention consists of prepending emx in front of the Bean name and appending Base on the base class name.

Out-of-the-box business logic always goes into the base JPO. This code is owned and managed by ENOVIA. Any extensions should be placed into the derived JPO where it will be automatically preserved when new releases of the applications are installed.

For extensions to be picked up, all references to JPO methods made by the system (either from Beans, triggers, or dynamic UI components) go through the derived JPO. If there are no extensions in the derived JPO, the base JPO functionality is picked up through inheritance.



Extending a JPO Method

The basic way to extend a JPO method is to write a method in the derived JPO that exactly matches the signature of an existing method in the base JPO.

The derived method can call the base method (using super.methodName()) if you want to have the method add additional logic to what ships out of the box. If the derived method never calls the base method, the result is that the derived method completely replaces what ships out of the box.

Important: Avoid copying code from the base JPO, instead call the base method and place custom logic in the derived JPO.

Use the derived JPO technique to ensure your customizations are preserved when new releases, service packs, and hot fixes are installed. The key is to isolate your modified logic into the derived JPO (as opposed to modifying the OOTB code directly). The installation routines do not copy derived JPOs into the database, if they already exist so there is no chance of losing your modifications.

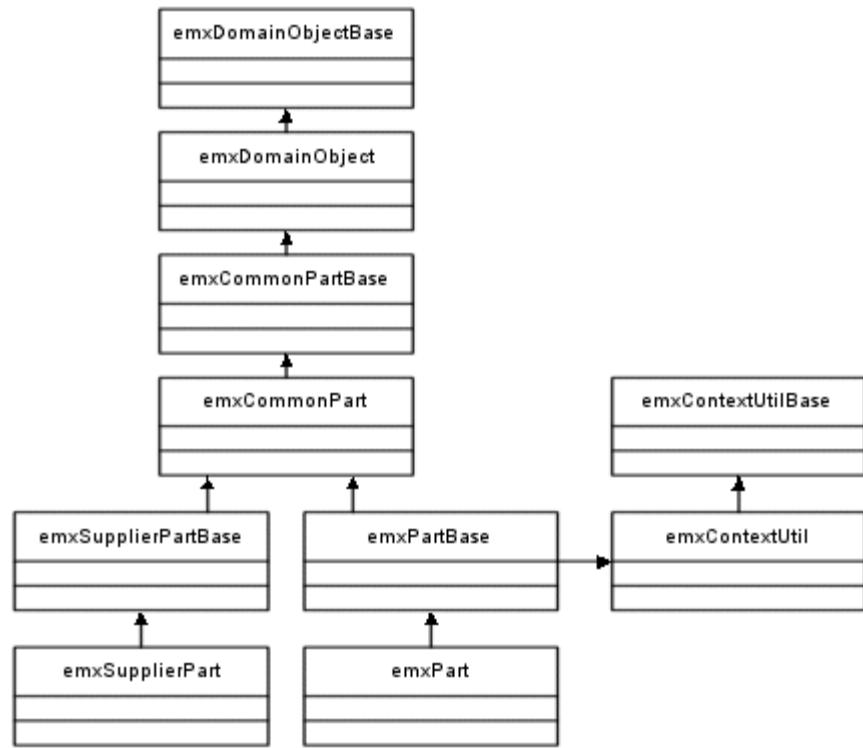
Methods called in the base JPO may have code changes, but their APIs remain consistent, so there should never be a case where your derived JPO no longer compiles after installing a new release. However, you may want to determine if any of the JPO methods you have extended changed in the base JPO and decide what impact these changes might have on your customization. If you copied code from a base JPO (not recommended), you may have to manually update the code.

The source code of your derived JPO should always be backed up in a source control system.



JPO Hierarchy

This section shows an example JPO hierarchy.



To be consistent with the existing naming conventions, **emxPartBase** should really be **emxEngineeringPartBase**, and **emxPart** should really be **emxEngineeringPart**. For historical reasons, some JPO names are named different than the current convention.

Extending Beans

The need to extend Beans decreases as more and more business logic is moved from the Beans into JPOs. However, when you have to extend a Bean, there is a mechanism in place that allows you to derive a Bean from an existing Bean and have the system use the derived Bean.

You can write your own class within an application using the same name as the common class and have your class extend the common class. Since these classes share the same name, take special care on import statements to avoid wildcards. Otherwise you may need to fully qualify the class references throughout the code. ENOVIA recommends that you use the wildcard on import statements for your application classes, but avoid the wildcard on import statements for common components classes.

- [Mapping File](#)
- [Extending a Bean Method](#)
- [The Search Path](#)

Mapping File

After defining your derived Bean class, you need to change the entry in the appropriate mapping file to indicate that newInstance should serve up an instance of your class instead of the out-of-the-box class.

For example, suppose you extend the Part Bean in the engineering.jar file and name your new class com.acme.apps.engineering.AcmePart. The entry:

```
type_Part=com.matrixone.apps.engineering.Part
```

should be changed to:

```
type_Part=com.acme.apps.engineering.AcmePart
```

The modified mapping file can then be placed back into the jar file from which it came or it can be placed in the properties directory in the application server (since these are searched ahead of the jar files).



Extending a Bean Method

The basic way to extend a Bean method is to write a method in the derived Bean that exactly matches the signature of an existing method in the base Bean.

The derived method can call the base method (using super.methodName()) if you want to have the method add additional logic to what ships out-of-the box. If the derived method never calls the base method, the derived method completely replaces what was originally shipped.

To extend a Bean:

1. Write a new class that extends the Bean (that is, derives from the out-of-the-box Bean found in one of the jar files). You can place the compiled code for this new class anywhere that is defined on the classpath.
2. Update the mapping file that specifies which Bean to use for a given business type or relationship type. The WEB-INF/classes directory is always searched before WEB-INF/lib (per Java spec), so this is a good place for Bean code to reside.

Each package has a mapping file where each Bean in that package is registered. For example, the Part Bean lives in both common and engineering. There is an entry in each mapping file.

- In engineering: type_Part=com.matrixone.apps.engineering.Part
- In common: type_Part=com.matrixone.apps.common.Part.

Each bean maps to the appropriate package.



The Search Path

The method DomainObject.newInstance() searches for the appropriate Bean based on the given type or object id.

1. If an application is given, the search begins in that application's mapping file. While searching for a match, the type hierarchy found in the schema is used to return the closest match.
2. If no match is found, the search moves to the common mapping file. If no match found, the method returns a DomainObject. For example, if no BracketPart or Part is found in the Engineering Central mapping file (say, for example it was removed), then the common mapping file would be searched (and if no BracketPart is in there, it would return a common Part).
3. If the common mapping file had no Part entry, a DomainObject is returned.

Eventually there will be a DomainObject.SYMBOLIC_type_BracketPart constant.

The key is to isolate your new classes and modified mapping file from being written over by new changes from ENOVIA. The installations do not install mapping files in the WEB-INF/classes directory so there is no chance of a clash. However, you will want to extract a new mapping file from the newly deployed jar file in case there have been modifications made since the

prior release. It should be straight forward to update your mapping file and place the changed file back into the WEB-INF/classes directory.

Extending JSPs

Java filters can be used to handle extensions to JSPs to preserve them across new releases of the ENOVIA products.

JSP filters let you intercept requests based on a URL pattern and perform alternative logic. Filters can be chained, which allows the results of one filter to be passed as input to a subsequent filter. To do this, place a copy the JSP in a special directory. This leaves the original JSP intact in its original directory.

This section presents these topics:

- [JSP Filters](#)
- [Resources](#)

JSP Filters

JSP Filters were first made available in J2EE application servers which implement the Servlet 2.3 specification.

The following table shows the minimal version of each application server.

Server Name	Minimum Version
WebLogic	6.1
WebSphere	5.0
SunOne	7.0

The ENOVIA Live Collaboration Server contains the CustomFilter class. This filter has been designed to look for a JSP:

1. With the same name as the one found in the current URL, but located in a subdirectory named custom (by default).
2. With the name defined by the ematrix.customDirName parameter.
3. With a name that is the concatenation of a specified prefix (with ematrix.customPrefix) and the name in the requested URL.

The filter information is configured using web.xml (found in the WEB-INF directory). The class is commented out in the web.xml file shipped with the ENOVIA products.

If you uncomment this class, filtering is activated and any custom JSPs takes control. To restore the system to its original out-of-the-box state, just comment out this section and restart the server.

You can turn on just the URL pattern of interest to improve performance since the filter only comes into play when the appropriate url pattern is matched. All other URLs pass through the system in a normal fashion (that is, with no filtering).

Refer to the *ENOVIA Live Collaboration Installation Guide* for details on configuring filtering.

The use of CustomFilter will ensure your customizations are preserved during the installation of new releases, service packs, and hot fixes.

- One option of CustomFilter places all of your modified JSPs in a separate directory. This makes it easy to backup your code and manage your code during upgrades. The downside is the need to modify paths in the JSP.
- The other option relies on a common prefix associated with your custom JSPs. With a simple grep-like command, you can easily find the custom pages from among the OOTB pages. New JSPs dropped into the system during an upgrade will never overwrite your custom JSPs. However, you will have to manually merge changes from the new OOTB JSPs back into your custom JSPs. One way to simplify this would be to diff the incoming JSPs with the originals to isolate which JSPs have changed and what those changes were. This delta change could then be analyzed and a determination made on how to incorporate the changes into your custom JSPs.



Resources

Java filters can transform a Request, or modify a Response to the items listed in this section.

Java filters can be used to:

- intercept a servlet's invocation
- examine a request before a servlet is called
- modify the request headers and request data
- modify the response headers and response data.

See "Taming your Tomcat: Filtering tricks for Tomcat 5" for an example of how to use filters in the most recent Servlet 2.4 specification. <http://www-106.ibm.com/developerworks/java/library/j-tomcat2/#N400061>.

About Writing JSPs

This section contains guidelines for writing JSPs for ENOVIA or custom applications.

The following topics are discussed:

- [Use Built-In Parameters to Make Pages Reusable](#)
- [Standard Error Handling](#)
- [JSP Layout for Error Handling](#)
- [Error Handling in the Same Request Scope](#)
- [Error Handling Not in the Same Request Scope](#)
- [Reset Context Limitation](#)

Use Built-In Parameters to Make Pages Reusable

The dynamic user interface components automatically append a number of parameters to URLs.

For example, the parameters emxSuiteDirectory, objectId, and ParentOID are all passed automatically. For a list of automatically-appended parameters, see [Parameters Automatically Passed to URLs](#).

It is important that when JSPs are written to be pop-ups (even if they are called from custom pages initially) that they use these automatically-appended URL parameters and do not invent new ones. That way, when you call a custom page from one of the configurable pages (emxTable.jsp, emxTree.jsp), you do not have to make changes.

It is also important not to pass in data that you can get from the automatically-appended parameters. Passing extra data creates pages that cannot be used from a configurable page.



Standard Error Handling

This section describes the error handling mechanisms used for the configurable components.

Whenever an exception or context-specific error happens in a JSP, an error message displays as a JavaScript popup alert message box. If there are multiple errors within the page, all error messages are collected by the error object and displayed to user.



To achieve this, an Error Object class called FrameworkException is instantiated for every JSP. This class is part of the com.matrixone.framework.util package. The error object instance name is emxNavErrorObject. The instantiation of the object is done at the top include file, emxNavigatorTopErrorInclude.jsp. The error object is used in the processing page for adding the error messages and finally used at the bottom include page, emxNavigatorBottomErrorInclude.jsp, for displaying any error message. This bottom include page also has the MQL Notice Include file, emxMQLNotice.jsp, which alerts any MQL error message that occurs during the processing.

The Error object emxNavErrorObject has the request scope, so the new error object is available to every request. Hence the object is destroyed from memory after displaying the page.

When the error/exception occurs, the page can call the emxNavErrorObject object method addMessage() (on the class FrameworkException).

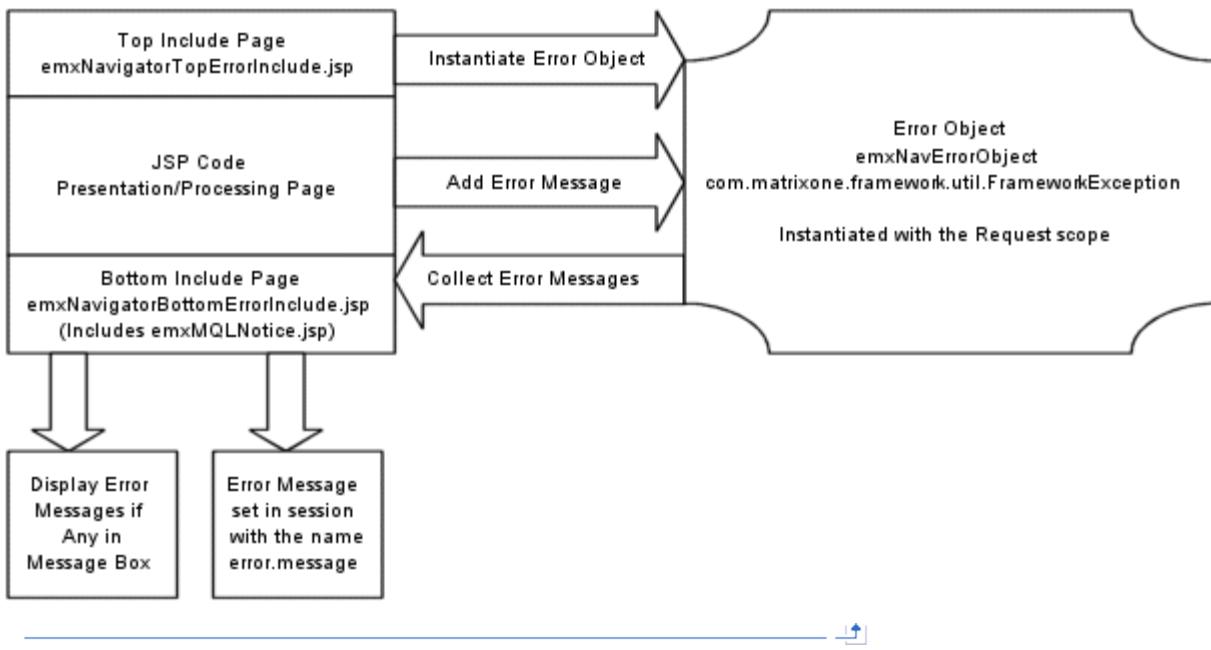
```
emxNavErrorObject.addMessage( "ERROR TEXT MESSAGE? . " );
```

Multiple error messages can be added to the error object and it maintains a list of messages within the error object. At the end of the page, the list of error messages is collected from the error object and displayed to the user in a single message box.



JSP Layout for Error Handling

Every JSP page that implements this error handling approach should have the following page layout.



Error Handling in the Same Request Scope

Every JSP page that implements error handling with the emxNavErrorHandlerObject must include the Top and Bottom Error Include files given in this section.

These are the top and bottom error include files:

- emxNavigatorTopErrorInclude.jsp
- emxNavigatorBottomErrorInclude.jsp

All the processing Java code within the JSP page (including jsp:forward and jsp:include) will be in the try.. catch.. finally block. If an error occurs in the processing code, an appropriate error message can be added to the error object using the addMessage() method. The addMessage method can be called within the page just after getting the error message or within the finally block.

The following sample skeleton code shows the syntax and method for implementing the error message. The instance name of the error handling class FrameworkException is emxNavErrorHandlerObject, which is instantiated in the Top Include.

```

// JSP code?
String emxErrorString = "";
try {
?
?
    // Access check
    ?..
    ?..
    emxNavErrorHandlerObject.addMessage("User does not have access to this object..");
?..
?..
} catch ( Exception e) {
?
?
    emxErrorString = (ex.toString()).trim();
} finally {
    emxNavErrorHandlerObject.addMessage(emxErrorString);
}

```

Error Handling Not in the Same Request Scope

If an error occurs in a pure processing page that is not within the same request scope of the referring page, you can set a session attribute to implement error handling.

When there is an error message in the processing page that needs to be displayed to the user, set a session attribute with name error.message with the value assigned to the ERROR TEXT to be displayed. For example:

```
session..setAttribute("ERROR TEXT MESSAGE? .");
```

If the JSP that displays subsequent to the processing page has implemented the error handling approach with

emxNavErrorObject, the session error message set in the session displays. This subsequent JSP page must follow the JSP layout given in [JSP Layout for Error Handling](#).



Reset Context Limitation

The Context.resetContext method should NOT be the first server call in a new JSP. It must be preceded by ANY server call with the new frame context such as the one in the example in this section.

```
MQLCommand.executeCommand(ctx, "print context").
```

To allow multi-threaded JSPs (for example, JSPs representing multi-frame browser pages), ENOVIA Live Collaboration creates a distinct frame context for each frame in each JSP. Each frame context is associated with the parent context via the application server session id. They are kept unique by appending "mxXXXXXX" to that session id.

It is critical for ENOVIA Live Collaboration to distinguish these different frame context objects to avoid operations performed simultaneously on multiple threads from interfering with each other. However, it is also important to correctly associate each of these frame contexts with the correct parent context to guarantee correct access checking against the business model.

Any other server call (as the first call on a page) will associate the frame context correctly with the parent, so a subsequent call to Context.resetContext will recognize the business administrative status, and allow the reset to a different context.

Configurable Pages and Commands

Business Process Services provides components you can use to build your applications. This section describes how to use configurable pages that let users perform specific tasks, such as viewing object history or choosing an object type.

In this section:

- [Configurable Preference Pages](#)
- [Configurable History Page](#)
- [Configurable Type Chooser](#)
- [Date Chooser](#)
- [Lifecycle Page](#)
- [Logout Command](#)
- [Change Password Command](#)
- [Page History Page and Command](#)
- [Quick File Access Details Table](#)
- [Context-Sensitive Help](#)

Configurable Preference Pages

This section describes how to configure and build preference pages. These pages let users specify general preferences for all applications and preferences specific to an application.

In this section:

- [About Preferences](#)
- [Adding a Preference Page](#)
- [Implementing a Currency Conversion](#)
- [Defining a Conversion Rate Preference](#)

About Preferences

The Tools menu on the global toolbar contains a Preferences link.

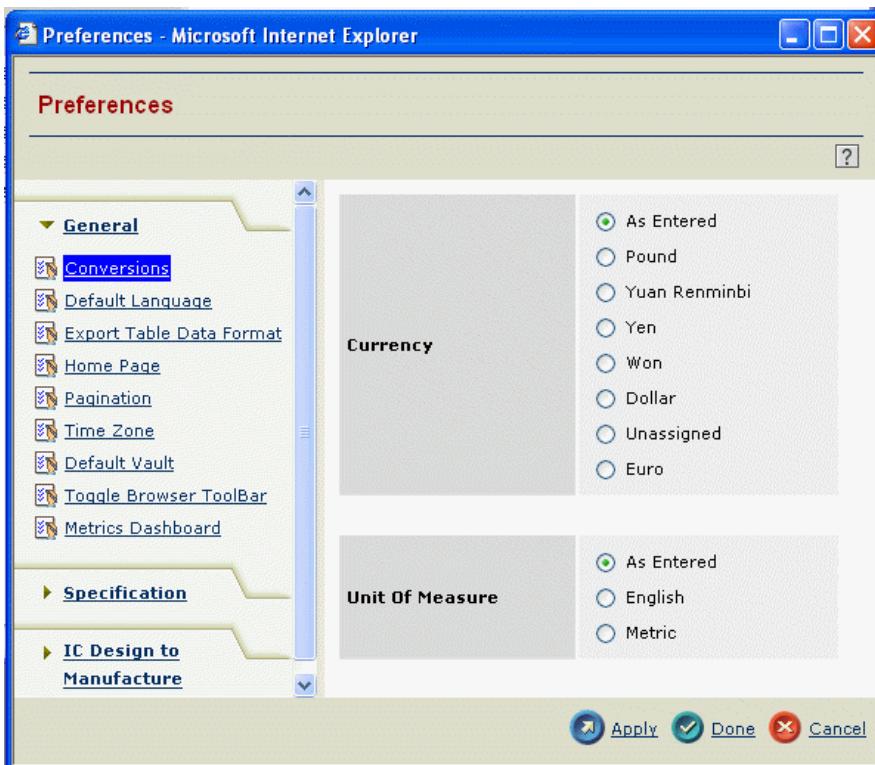
The following topics are discussed:

- [The Preferences Dialog Box](#)
- [Mechanisms that Define Preferences](#)
- [About Currency Conversions](#)
- [About Unit of Measure Conversions](#)
- [Default Language for System Notifications](#)
- [Export Table Data Format](#)
- [Home Page Preference](#)
- [Toggle Browser Toolbar](#)

The Preferences Dialog Box

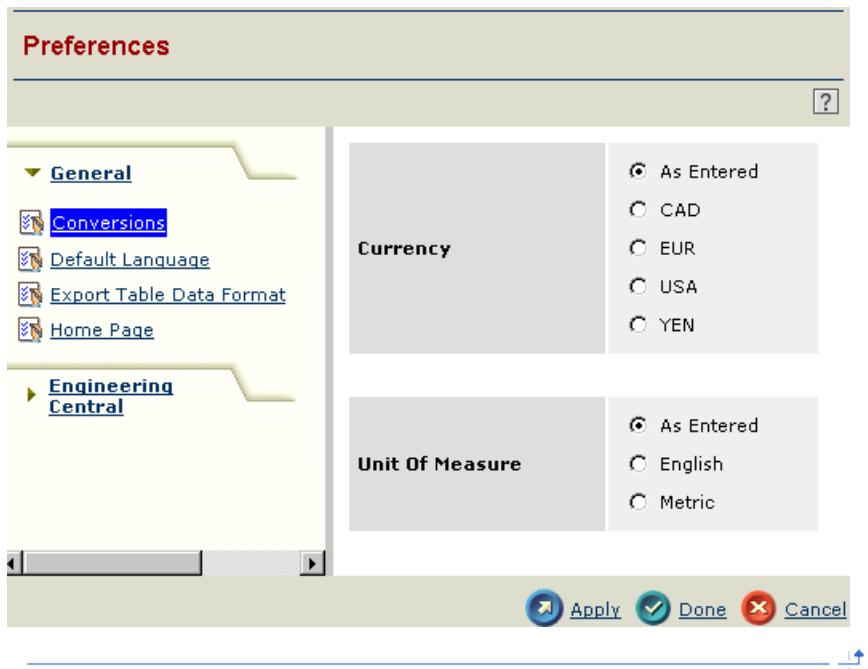
When a user clicks the Preferences link, the Preferences page appears in a popup window.

The tools menu is accessed by clicking  on the global toolbar.



Links for specific preferences can be categorized based on whether they apply to multiple applications, general applications, or to a specific application. Business Process Services installs with a number of general preferences and specific ENOVIA products may install additional general or application-specific preferences.

These preferences can be configured and custom preferences can be added. For example, below is a Preferences page with preferences added for Engineering Central.



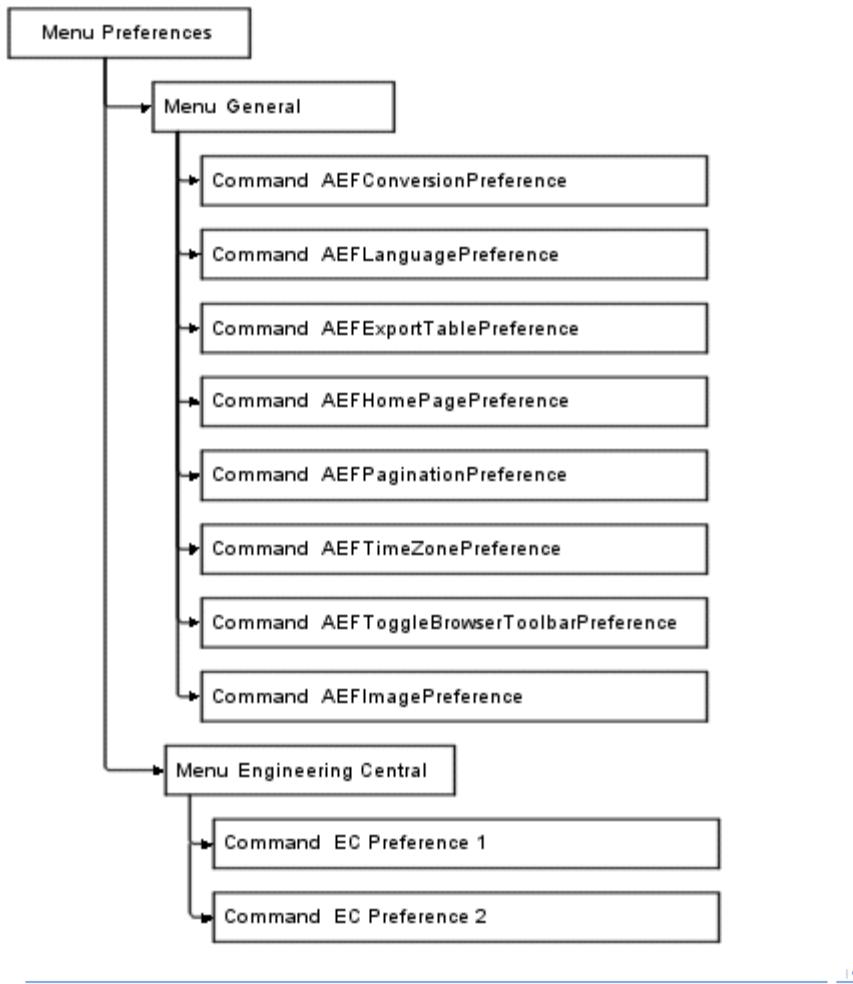
Mechanisms that Define Preferences

Business Process Services installs an administrative menu object, called Preferences, which represents the Preferences page.

Categories of preferences, such as General and Engineering Central, are defined using menu objects assigned to the Preferences menu. For example, to define the General category for preferences, Business Process Services installs with a menu object called General, which is assigned to the Preferences menu.

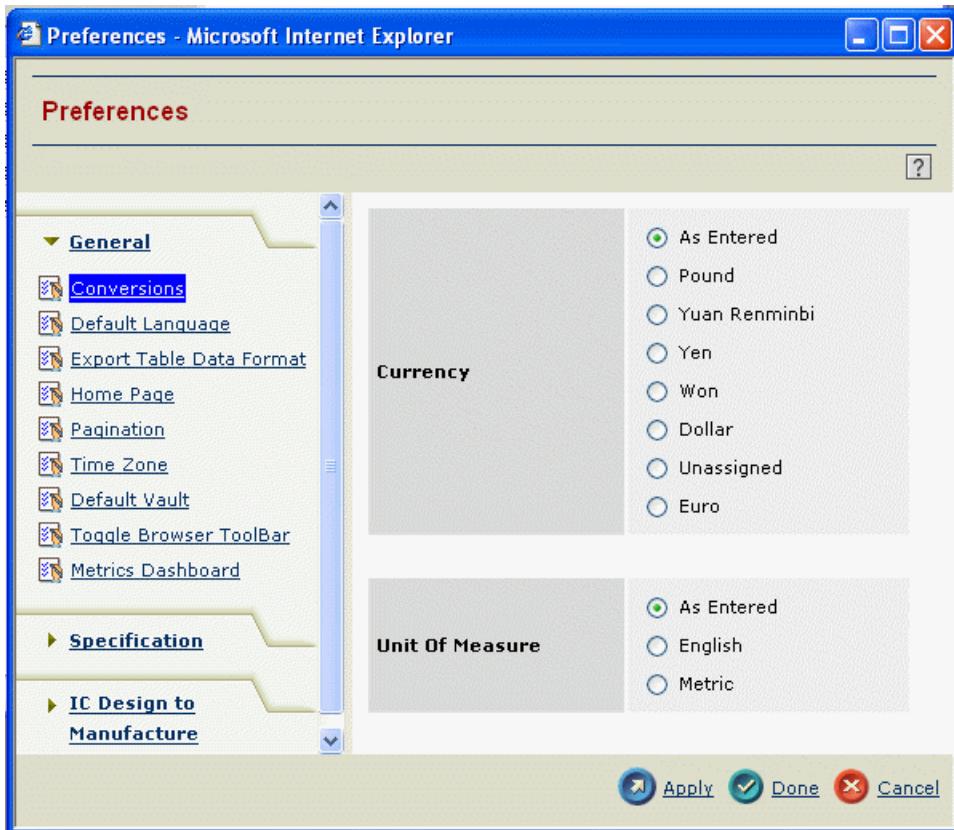
The specific preferences within a category are defined using command objects which are connected to the menu object for the category. For example, to define the link for the Home Page Preference page, the framework installs with a command called AEFHomePagePreference. This command is assigned to the General menu.

This graphic shows a sample administrative object structure needed for an installation that uses the General preferences installed with the framework and two additional preferences for Engineering Central. All the objects shown here are installed Business Process Services except the menu and two command objects for Engineering Central. These objects might be installed with Engineering Central or could be added as custom preferences.



About Currency Conversions

This preference lets users choose the currency in which they want to view monetary attributes, such as unit price. This image shows the default currency conversion preferences.



When at least one column in a configurable table is defined with the setting format=currency, the table page includes a Conversion tool in the page toolbar. (The tool also displays if a column is configured with format=UOM, for unit of measure conversion.) Applications may also include the Conversion tool in the page toolbar for pages that are not built using the configurable table. Like the configurable Conversion tool, these non-configurable instances of the Conversion tool operate using the currency selected in preferences, as described below.

When a user opens a page that has column data configured as format=currency, all currency data is listed as the data was entered. When the user clicks the tool, the page opens in a new window and all currency columns are converted to the currency chosen on the Currency and Unit of Measure Preferences page, shown above (unit of measure columns are also converted). If the default selection of As Entered is selected as the preferred currency, no conversion is made.

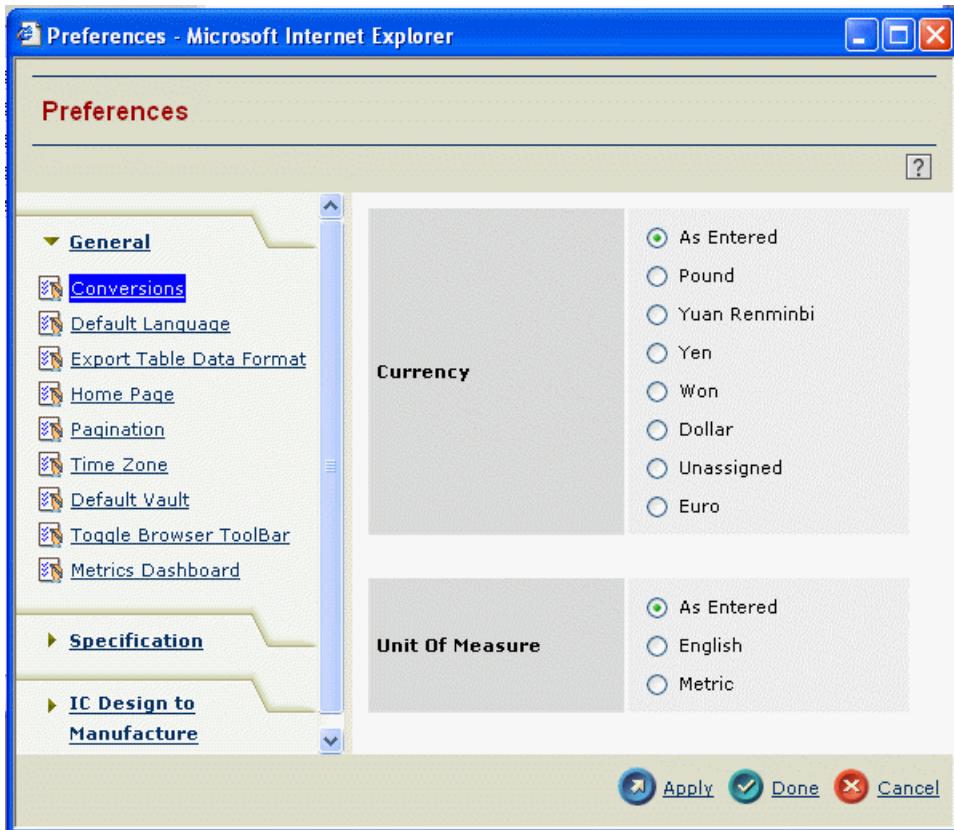
This table summarizes the configuration options for currency conversion preferences.

Preference	Available choices defined by:	Default value defined by:	User's selection stored:
Currency conversion	Range values on the Currency attribute	Specifying a default range value for the Currency attribute	for global use in current session

About Unit of Measure Conversions

This preference lets users choose the unit of measure in which they want to view measurement attributes, such as Weight. For example, if a user enters weight data using the Metric unit grams, another user can convert this data to the English unit pounds. Each English unit can be converted to only one Metric unit. Currently, only Sourcing Central uses currency and unit of measure conversions.

The Unit of Measure conversion is a separate functionality than the units of measure provided by the Dimension administrative object, although the preference set by the user controls which units within a dimension are displayed as described in [Table Column Data Definition](#).



When at least one column in a configurable table is defined with the setting format=UOM, the table page includes a Conversion tool in the page toolbar. (The tool also displays if a column is configured with format=currency, for currency conversion.)

When a user opens a page that has column data configured as format=UOM, all measurement data is listed as the data was entered. When the user clicks the tool, the page opens in a new window and all measurement columns are converted to the unit of measure chosen on the Currency and Unit of Measure Preferences page, shown above (currency columns are also converted). If the default selection of As Entered is selected as the preferred unit of measure, no conversion is made.

This table summarizes the configuration options for this Preference.

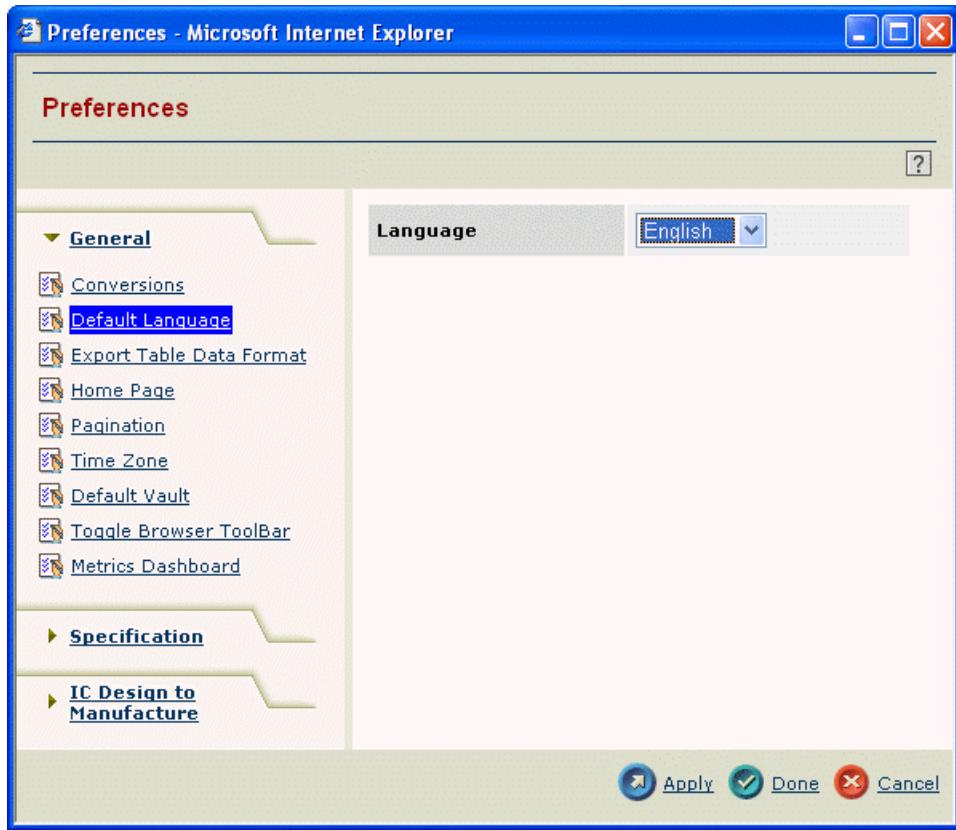
Preference	Available choices defined by:	Default value defined by:	User's selection stored:
Unit of measure	Range values on the Unit of Measure attribute	Specifying a default range value for the Unit of Measure attribute	for global use in current session



Default Language for System Notifications

This preference lets users choose the language in which they want to receive system notifications.

Some applications include an equivalent preference setting on the person profile page. For example, in Common Components, the page for editing a person's profile information includes a setting called System Generated Mail Preference. This is the same setting as the Default Language setting in Preferences. If the user chooses a particular language from Preferences, the field on the profile page shows the same language selected and vice versa.

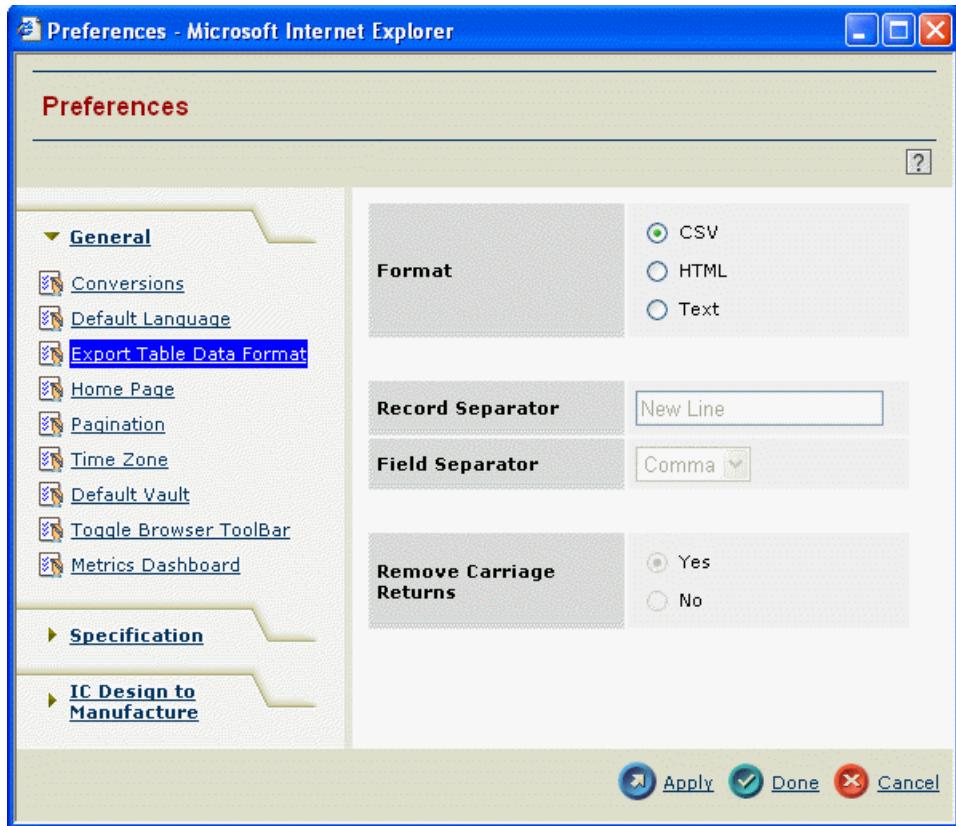


This table summarizes the configuration options for this preferences page.

Preference	Available choices defined by this property in emxSystem.properties	Default value defined in this property in emxSystem.properties:	User's selection stored:
Language for notifications	emxFramework.Preferences.Language.Choices = en, ja, it, fr, ge, zh_cn	emxFramework.Preferences.Language.Default = en	IconMailLanguagePreference property on the user's person administrative object

Export Table Data Format

This preference lets users choose the format they want to use when exporting table data. If the user chooses Text format, there are additional choices for separators and how to handle carriage returns are also used.



This table summarizes the configuration options for this preferences page.

Preference	Available choices defined by this property in emxSystem.properties	*Default value defined in this property in emxSystem.properties:	User's selection stored:
Export table data format	emxFramework.Preferences.ExportFormat.Choices = CSV, HTML, Text	emxFramework.Preferences.ExportFormat.Default = CSV	ExportFormat property on the user's person administrative object
Field separator (available only when export type is Text)	emxFramework.Preferences.FieldSeparator.Choices = Pipe, Tab, Comma	emxFramework.Preferences.FieldSeparator.Default = Comma	preference_FieldSeparator property on the user's person administrative object
Record separator (available only when export type is Text)	This is a text box so no options are available for selecting.	The default value is New Line	preference_RecordSeparator property on the user's person administrative object
Remove carriage returns (available only when export type is Text)	emxFramework.Preferences.RemoveCarriageReturns.Choices = Yes, No	emxFramework.Preferences.RemoveCarriageReturns.Default = Yes	preference_RemoveCarriageReturns property on the user's person administrative object
Multiple column value separator	If a column contains multiple values, each value will be separated by the value assigned to this property. The default is a new line (\n).		N/A

* The default value defines the selected value only if the user does not change that preference. The user's preference always overrides the default value.

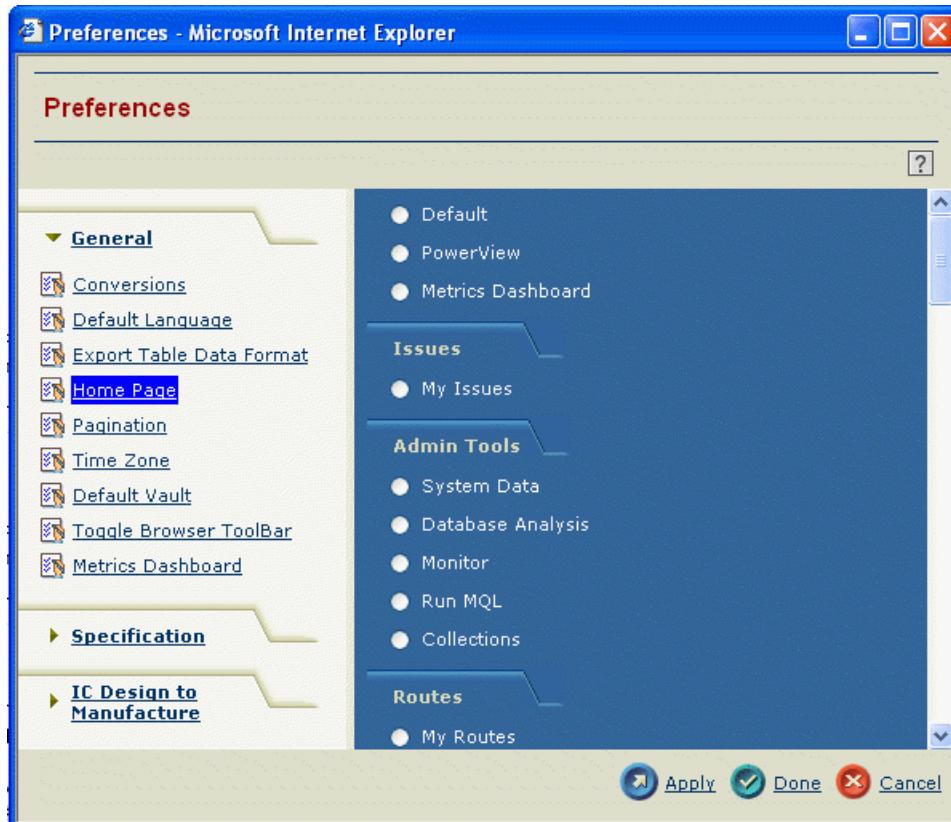
You can also define the value used to separate multiple values in a column using this property in emxSystem.properties. The

separator only applies to CSV and text formats, not to HTML. This is a system-wide setting that users cannot set through preferences.

```
emxFramework.Preferences.FieldValueSeparator.Delimiter = \n
```

Home Page Preference

This Preference lets users choose their home page. This is the page that appears when they first log in and then click the Home Page tool.



This table summarizes the configuration options for this preferences page.

Preference	Available choices defined by this property in emxSystem.properties	Default value defined in this property in emxSystem.properties:	User's selection stored:
Home Page	All commands on the user's My Desk (displays as) menu are listed, except those with Selectable in Preferences set to False.	Not applicable.	In two properties on the user's person administrative object: preference_Menu stores the application menu name and preference_Command stores the specific My Desk command within that application menu

Toggle Browser Toolbar

This preference lets users choose whether to show the browser's toolbar, or to hide it to provide more room for the ENOVIA products window.



This table summarizes the configuration options for this preferences page:

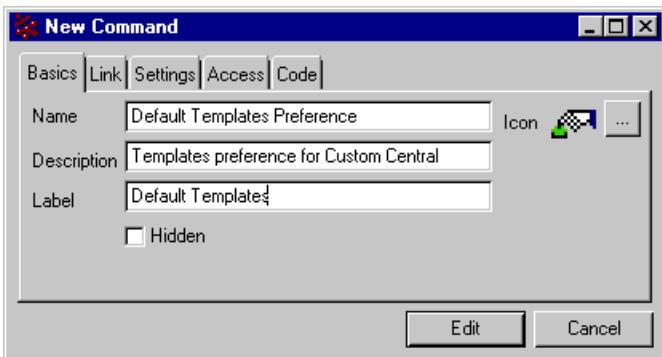
Preference	Available choices defined by this property in emxSystem.properties	Default value defined in this property in emxSystem.properties:	User's selection stored:
Toggle Browser Toolbar	emxFramework.Preferences.ToggleBrowserToolbar = true,false	emxFramework.Preferences.ToggleBrowserToolbar = true	preference_Toolbar property on the user's person admin object

True means to show the browser toolbar; false means to hide the browser toolbar.

Adding a Preference Page

This procedure lists the main steps for adding a Preference page and adding a new preference category to place the new page under, if needed. The graphics in the procedure illustrate the steps needed to create a Preference called Default Templates for the Custom Central application.

1. Launch Business Modeler.
2. To create the command for the new preference, follow these steps:
 - a. Click **Object > New > Command**.
 - b. Fill in the parameters (on the Basic tab) and settings (on the Settings tab) for the command as defined in the table.



Parameter/Setting	Description	Accepted Values/Examples
Label	The text displayed on the link for the preference. The link appears under the category name to which the command is assigned. Either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components . The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text.	Home Page Date/Time Format emxFramework.Preferences.Conversions
Name	Name of preference.	Home Page Default Language
href	The URL executed when the link for the preference is clicked. This URL is displayed in the right frame of the Preferences page. The value for the href parameter should be a JavaScript function or JSP and any associated parameters. You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components .	\${COMMON_DIR}/emxPrefConversions.jsp \${SUITE_DIR}/emxENCPrefVault.jsp
Access	The persons, roles and groups who can access the preference. To make the link available to all users, regardless of role/group assignments, choose All.	Names of group, role, person administrative objects. or All (default)
Setting: Image	Icon displayed in the Preference window.	COMMON_DIR/buttonToolbarPreferences.gif
Setting: Registered Suite	The name of a suite that registered this command	Framework TeamCentral

- c. Click **Create**.
3. To create a new category for the preference, follow these steps.

If the preference fits under an existing category (such as Engineering Central or another ENOVIA product), skip this step.

- a. Click Object > New > Menu.
- b. Fill in the parameters (on the Basic tab) and settings (on the Settings tab) for the Menu as defined in the table.

Parameter/Setting	Description	Accepted Values/Examples
Label	<p>The text for the category on the Preferences page. Either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components.</p> <p>The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text.</p>	Engineering Central Custom Central
Name	Name of category.	Engineering Central Custom Central
Setting: Registered Suite	The name of a suite that registered this command	Framework TeamCentral

- c. Click Create.

4. To add the new command to a menu, follow these steps: Assign the command you created in Step 1 to the appropriate category menu object. This is the menu object you created in Step 2 or an existing menu object.

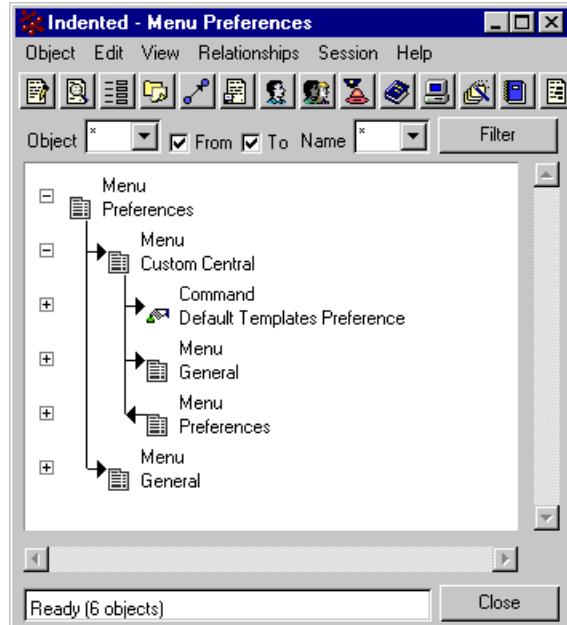
- a. Locate the needed menu, either the one you just created or an existing menu where you want to add the command for the new preference, and open it for editing.
- b. Click the Items tab and add the menu object you just created.
- c. Click Edit to save the menu.

5. If you created a new menu, follow these steps to add it to the Preference menu:

If you added the command to an existing menu, that menu should already be included in the Preference menu and you can skip this step.

- a. Open the Preferences menu object for editing.
- b. On the Items tab, add the menu you created above.
- c. Click Edit to save the Preferences menu.

The Preferences menu should show the new menu and/or command.



The preference is added to the list of preferences as shown in the image.

Preferences

[?]

General

- [Conversions](#)
- [Default Language](#)
- [Export Table Data Format](#)
- [Home Page](#)

Custom Central Preferences

- [Default Templates](#)

Currency

- As Entered
 CAD
 EUR
 USA
 YEN

Unit Of Measure

- As Entered
 English
 Metric

 [Apply](#)  [Done](#)  [Cancel](#)

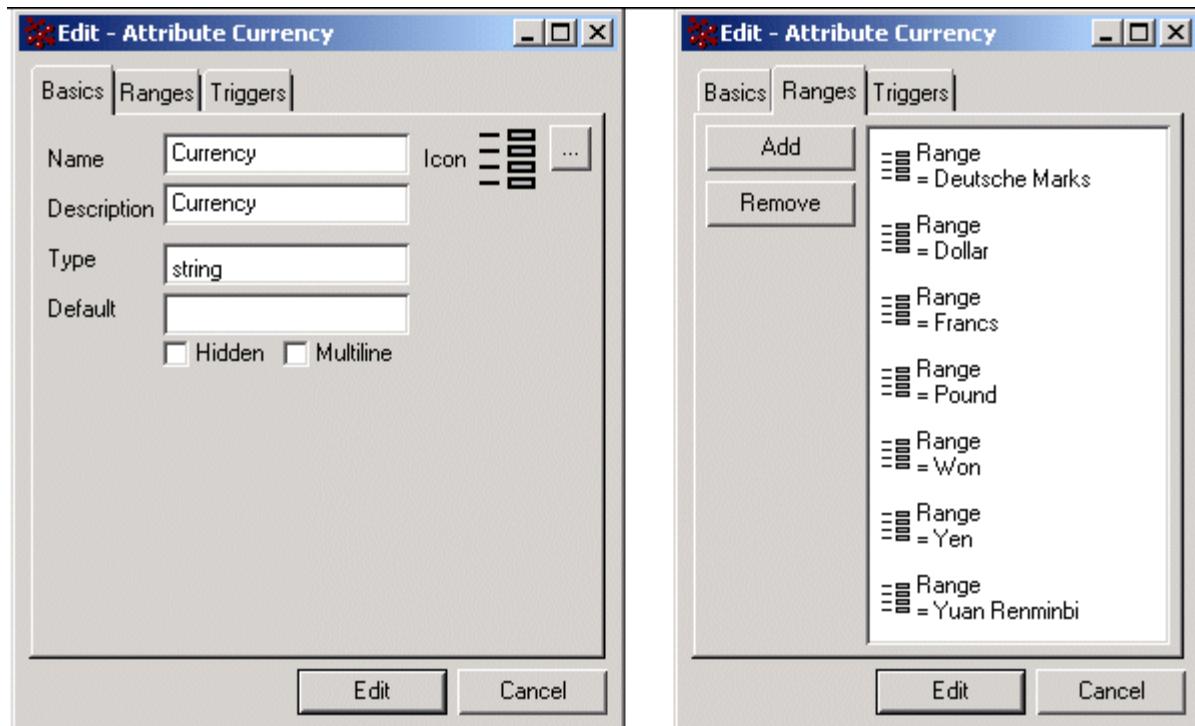
In this example, **Custom Central Preferences** is the new category (menu) for the Preferences page, and **Default Templates** is the new command.

Implementing a Currency Conversion

You can implement a custom currency conversion.

1. Define which currencies users can convert to, for example, you could choose to let users convert to Yen, Euros, and Dollars. In Business Modeler or MQL, add and remove range values for the Currency attribute so it includes all currencies that you want to convert. Use the standard abbreviations for each currency.

These Business Modeler dialog boxes show where currency ranges are defined.



2. Using the People and Organizations tool, define currency conversions for each company using the Currency Exchange Rates category in the company's category list. For instructions, see the Common Components User's Guide help or user guide.
3. In a configurable table, identify the currency columns that users should be able to convert by adding the format=currency setting for the column.
4. To display the table with the converted currency data in the new window, the table component uses the ConvertCurrencyTag TagLib. The TagLib uses these parameters to configure the displayed converted data.

TagLib Parameter	Column Setting	Description
from	Currency Expression	<p>The name of the currency to convert from.</p> <p>Setting "Currency Expression" is assigned to a select clause, which provides the value for the Currency attribute for a specific object or relationship. This should be the currency the user entered the data in (the As Entered currency).</p> <p>For example, to convert data for an RFQ Quotation, the following select clause returns the currency format.</p> <p>to[Supplier Line Item].attribute[Currency]</p>
to		<p>The name of the currency to convert to.</p> <p>The value for this "to" parameter is obtained from the session and is the preferred currency selected by the user.</p> <p>If not defined, no conversion is applied.</p>
value		Actual value to be displayed in the column (currency).
date	Effective Date Expression	The "Effective Date Expression" is assigned to a select clause, which provides the value for the Effectivity Date attribute on a specific object or relationship. The system

		<p>uses this date to get the currency conversion whose rate period falls within this date.</p> <p>For example, for data for an RFQ Quotation, the following select clause returns the effective date.</p> <pre><code>to[Supplier Line Item].attribute[Effectivity Date]</code></pre>
decimalSeparator		<p>Decimal separator symbol for displaying the currency value. This symbol is defined in emxSystem.properties using the key:</p> <pre><code>emxFramework.DecimalSymbol = .</code></pre>
digitSeparatorPreference		<p>Determines whether to display a thousandths separator for currency and quantity fields (for example, 1,000). This symbol is defined in emxSystem.properties using the key:</p> <pre><code>emxFramework.DigitSeparator = false</code></pre>

For example:

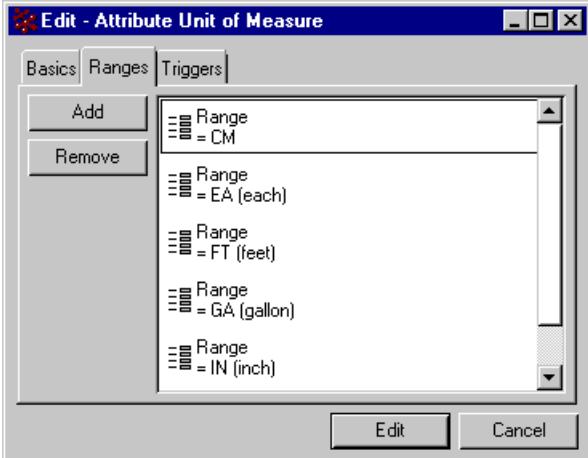
```
<framework:convertCurrency
    from="Dollar"
    to="Yen"
    value="50000"
    date="25/05/2001"
    decimalSeparator=". "
    digitSeparatorPreference="true"
```

Defining a Conversion Rate Preference

You can define conversion rates for English and Metric units. Perform the steps in this procedure for each English unit that you want users to be able to convert to a Metric unit.

1. In Business Modeler, define range values for each conversion rate you want to define:

- a. Open the Unit of Measure attribute for editing.



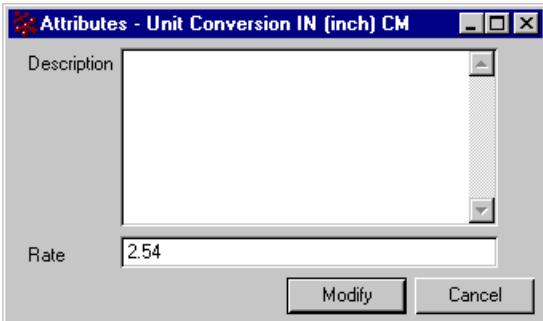
- b. Click the **Ranges** tab.
- c. For each new range value, click **Add** and enter a name.
- d. Click **Edit**.

2. To enter the conversion rates using Matrix Navigator or PowerWeb, follow these steps for each Unit Conversion.

- a. Click **Object > New > Original**.
- b. Select **Unit Conversion** as the type.

3. Enter a name that matches the English unit and the Revision matches the Metric unit. Both units must be added to the range values for the Unit of Measure attribute.

4. In the Rate field, enter the conversion rate between the English and Metric unit.



5. Click **Modify**.
6. To display the table with the converted measurement data in the new window, the table component uses Tag Lib. The following table lists column settings that need to be configured for Unit of Measure columns.

Taglib Parameter	Column Setting	Description
map		Table component obtains the conversion map from the system using the bean UnitConversion.
from	UOM Expression	<p>The name of the Unit of Measure from which to convert.</p> <p>The Unit of Measure Expression setting for the table column is assigned to a select clause, which provides the value for the Unit of Measure attribute for a specific object or relationship.</p> <p>For example, for data for RFQ Quotations, the following select clause returns the currency format.</p>

		to[Supplier Line Item].attribute[Unit of Measure]
to		<p>The name of the unit of measure to convert.</p> <p>The value for this "to" parameter is obtained from the session and is the preferred currency selected by the user.</p> <p>If not defined, no conversion is applied.</p>
value		Actual value to display in the column (measurement value obtained from database).
decimalSeparator		Decimal separator symbol that should be used for displaying the measurement value. This symbol is defined in emxSystem.properties using the key: emxFramework.DecimalSymbol = .
digitSeparatorPreference		Determines whether to display a thousandths separator for currency and quantity fields (for example, 1,000). This symbol is defined in emxSystem.properties using the key: emxFramework.DigitSeparator = false

For example:

```
<framework:convertUnit
    map = <%=UnitConversion.getUnitConversionMap(context)%>
        from="IN (inch)"
        to="English"
        value="22323"
        decimalSeparator=". "
        digitSeparatorPreference="true"
```

Configurable History Page

The framework installs with a configurable History page called emxHistory.jsp. The History page lists the actions that have occurred for the current business object. This page can be configured for your company's business needs.

In this section:

- [About Configurable History Pages](#)
- [Parameters Used by emxHistory.jsp](#)
- [Including Configurable History in a Custom Application](#)

About Configurable History Pages

Business Process Services installs with a configurable History page called emxHistory.jsp. The History page lists the actions that have occurred for the current business object.

The following topics are discussed:

- [Configurable Options for History](#)
- [Preconfigured History Command Objects](#)

Configurable Options for History

By passing parameters to the JSP, the page can be configured as described in this section.

- display events for the current revision only or for all revisions
- label the revisions "revisions" or "versions", when configured to display events for all revisions
- show only specific action types
- show or hide the action type filter
- show or hide the text filter

This figure shows the History page as it is installed out-of-the-box.

Test Everything: History

Action Type *	...	and *	Filter	?
Jan 16, 2009 2:32:13 PM	creator	Create	Inactive	revised from:
Jan 16, 2009 2:32:13 PM	creator	Override	Inactive	
Jan 16, 2009 2:32:13 PM	creator	Promote	Active	
Jan 16, 2009 2:32:13 PM	creator	Connect	Active	Employee from Company Company Name -
Jan 16, 2009 2:32:15 PM	creator	Modify	Active	Last Name: Everything was: Test Everything
Jan 16, 2009 2:32:15 PM	creator	Modify	Active	First Name: Test was: Test Everything
Jan 16, 2009 2:32:24 PM	creator	Connect	Active	Company Representative from Company Company Name -
Jan 16, 2009 2:32:32 PM	creator	Change Vault	Active	vault: eService Production was: eService Administration
Jan 16, 2009 2:32:40 PM	creator	Connect	Active	Member from Company Company Name -

Preconfigured History Command Objects

Business Process Services installs with two commands that can be used by applications to call the History page. Both of these commands and the configuration of the History page that each command calls is described below.

Both commands are configured to display the Action Type and text filters and have no prefilter action types.

Command	Command
AEFHistory	AEFHISTORYAllVersions

The AEFHistory command calls the History page with a parameter set so only the actions for the current revision of the business object are listed. The History page always includes the ability for users to filter the list by action type and by specific text strings in any column other than action type. This command is designed to display the History page in the content frame and therefore should be used as a tree category. This screen shows the current revision mode of the History page.

Molded Part CM-202030-01: History

Action Type and

Date	User	Action	Action Details	State	Message
Aug 9, 2002 3:16:36 PM	Test Everything	connect	Design Responsibility from Department Mechanical Design Company Name Mechanical Design	Release	
Aug 9, 2002 3:16:24 PM	Test Everything	connect	New Part / Part Revision from ECO ECO	Release	
Aug 9, 2002 3:16:17 PM	Test Everything	override		Review	
Aug 9, 2002 3:16:17 PM	Test Everything	promote		Release	
Aug 9, 2002 3:16:16 PM	Test Everything	override		Preliminary	
Aug 9, 2002 3:16:16 PM	Test Everything	promote		Review	
Aug 9, 2002 3:16:06 PM	Test Everything	connect	Part Family Member from Part Family CM	Preliminary	
Aug 9, 2002 3:16:06 PM	Test Everything	connect	EBOM from Mechanical Part CM	Preliminary	
Aug 9, 2002 3:16:01 PM	Test Everything	create		Preliminary	revisioned from:
Aug 22, 2002 10:49:08 AM	Test Everything	revisioned		Release	revision: Molded Part CM-202030-01 2

The AEFHistoryAllVersions command calls a History page with a parameter set so actions are listed for all versions of the object. When the History page includes all versions of the object, it also includes drop-down lists in the upper right corner that let users specify the versions they want to see. This command is designed to display the History page in a popup window and therefore should be used as a toolbar item. This screen shows the all revisions mode of the History page.

Molded Part CM-202030-01: History

From Ver: to Ver: Refresh

Action Type ... and



Version 1

Date	User	Action	Action Details	State	Message
Aug 9, 2002 3:16:36 PM	Test Everything	connect	Design Responsibility from Department Mechanical Design Company Name Mechanical Design	Release	
Aug 9, 2002 3:16:24 PM	Test Everything	connect	New Part / Part Revision from ECO ECO	Release	
Aug 9, 2002 3:16:17 PM	Test Everything	override		Review	
Aug 9, 2002 3:16:16 PM	Test Everything	promote		Review	
Aug 9, 2002 3:16:06 PM	Test Everything	connect	Part Family Member from Part Family CM	Preliminary	
Aug 9, 2002 3:16:06 PM	Test Everything	connect	EBOM from Mechanical Part CM	Preliminary	
Aug 9, 2002 3:16:01 PM	Test Everything	create		Preliminary	revisioned from:
Aug 22, 2002 10:49:08 AM	Test Everything	revisioned		Release	revision: Molded Part CM-202030-01 2

Version 2

Date	User	Action	Action Details	State	Message
Aug 22, 2002 10:49:12 AM	Test Everything	modify		Preliminary	Production Make Buy Code: Buy was: Make
Aug 22, 2002 10:49:12 AM	Test Everything	connect	Part Family Member from Part Family CM	Preliminary	

Parameters Used by emxHistory.jsp

The emxHistory.jsp page accepts the parameters listed in this table. Additionally, emxHistory.jsp requires the objectId parameter so it knows which object to get history for. The objectId parameter is automatically passed to the page from the tree in the Current Revision mode and from the toolbar item in the All Revisions mode.

Parameter	Description	Accepted Input Values
Header	The text to use for the header of the History page. This parameter can either be a string resource ID or a mixture of macros and text or simply text.	\$<type> \$<name> \$<revision> emxFrameworkStringResource.Common.HistoryPageHeading Object History
HistoryMode	Determines whether the entire revision chain or just the current revision history should be displayed.	AllRevisions--The page displays the entire revision history of the object. CurrentRevision (default)--The page displays the history for the particular revision that is being reviewed. For example, a part may have REV A, Rev B and Rev C. If we are reviewing REV B and the HistoryMode is set to "CurrentRevision", only the history for REV B is presented. If HistoryMode is set to "AllRevisions", then the history for REV A, B, C is presented.
preFilter	<p>Determines which action types to display when the History page comes up. If the parameter is not passed, the page lists all action types.</p> <p>If the Action Type filter control is shown on the page (determined by the ShowFilterAction parameter), the preFilter actions are listed in the text box and the filter is applied to the history list when the page first opens. The user can use the Action Type filter to change the filter, adding and removing displayed actions as needed.</p> <p>If the Action type filter control is not shown, the user cannot change the preFilter actions. By setting preFilter actions and hiding the Action Type filter, you can hide specific actions that you do not want users to see.</p>	<p>The parameter accepts two types of values:</p> <p>Comma delimited list of action types: <code>preFilter=connect, disconnect, create</code></p> <p>String Resource ID: <code>preFilter=emxSystem.preFilter.List</code></p> <p>To specify which string resource properties file to look in to get the value, include the SuiteKey parameter. If the SuiteKey is not passed in, the History page looks for the key in the emxSystem.properties file. For example, in the emxSystem.properties file, the value for the emxSystem.preFilter.List might be "connect, create, disconnect".</p>
ShowFilterAction	Determines whether to display the action type filter or not. 	true (default)--The action type filter displays. false--The action type filter does not display.
ShowFilterTextBox	Determines whether to display the filter text box or not. 	true (default)--The filter text box displays. false--The filter text box does not display.
subHeader	Only needed when using All Revisions mode. Determines whether to display Version or Revision for the SubHeader, which separates the history events for each revision.	Version--The History page looks in the emxFrameworkStringResource.properties file for value of the emxFramework.History.Version property. Revision--The History page looks in the emxFrameworkStringResource.properties file for value of the emxFramework.History.Revision property.
SuiteKey	Determines which string resource property file to look in for the preFilter property key. This parameter is only used when the preFilter parameter is passed in as a resource ID. For example, SuiteKey=eServiceSuiteEngineeringCentral would be required for the preFilter	Possible values for the SuiteKey parameter are: eServiceSuiteTeamCentral, eServiceSuiteEngineeringCentral, eServiceSuiteProgramCentral, or any other suite name.

parameter whose value is in the
EngineeringCentral properties file,
emxEngineeringCentral.properties.

Including Configurable History in a Custom Application

This simplest way to use the History page in an application is to use one of the command objects installed with the framework.

1. Assign the command object (AEFHistory or AEFHistoryAllVersions) to the appropriate menu object.

For example, to use the AEFHistory object, assign it to the menu object that represents the tree you want to add the History category to. To use the AEFHistoryAllVersions objects, assign the object to the menu object for the toolbar item.

2. To configure your own History command object:

- a. In the href parameter of the administrative command object that will call the History page, enter the emxHistory.jsp, plus the needed parameters (see [Parameters Used by emxHistory.jsp](#)) and a path indicator

For example:

```
 ${COMMON_DIR}/  
emxHistory.jsp?HistoryMode=AllRevisions&Header=emxFrameworkStri  
ngResource.Common.HistoryPageHeading&subHeader=Version&preFilte  
r=emxEngineeringCentral.history.preFilterList&SuiteKey=eService  
SuiteEngineeringCentral  
  
 ${COMMON_DIR}/  
emxHistory.jsp?HistoryMode=CurrentRevision&Header=$<type>  
 $<name>  
 $<revision>&subHeader=Revision&ShowFilterAction=true&ShowFilter  
 TextBox=false&preFilter=connect,disconnect,promote
```

- b. To configure a toolbar item that opens the History page in a popup window, enter these settings:

- Target Location=popup
- Row Select=none
- Registered Suite=Framework

- c. To configure a tree category that opens the History page in the content frame, enter these settings:

- Target Location=content
- Image=IMAGE_FILENAME
- Registered Suite=Framework

- d. Connect the command object to the menu administrative object that will contain it. For example, if the command object is for a toolbar item, connect the command to the menu object for the Actions menu. If it is a tree category, connect it to the menu object for the tree.

Configurable Type Chooser

Business Process Services installs with a configurable Type Chooser called emxTypeChooser.jsp. The Type Chooser lists types defined in the database and lets users choose from the list.

In this section:

- [!\[\]\(eb9d9bab5872f792b1d488e68d57e314_img.jpg\) About the Configurable Type Chooser](#)
- [!\[\]\(1da41556429070f07c047bfe3d7f4357_img.jpg\) Parameters Used by the Type Chooser](#)
- [!\[\]\(e196cb797861b3cf10e296a34d9e93ad_img.jpg\) Calling the Configurable Type Chooser from a Page](#)

About the Configurable Type Chooser

Many business processes within the ENOVIA products require that users specify a type for a business object. When searching for objects, users can often narrow the search to specified types. The framework installs with a configurable Type Chooser called emxTypeChooser.jsp. The Type Chooser lists types defined in the database and lets users choose from the list. The Type Chooser contains text filter fields and a hierarchical tree structure that helps users find the type(s) for which they are looking.

The following topics are discussed:

- [How the Type Chooser Works](#)
- [Internationalized Selected Type Names](#)
- [About the Page \(Configurable or JSP\) that Uses a Type Chooser](#)

How the Type Chooser Works

The Type Chooser can be called from a field defined on a configurable Form page or from any standard JSP.

Important: Wherever feasible, you should take advantage of the automatic type ahead chooser. See [Automatic Type Ahead](#). This pre-configured chooser allows users to start typing in the field, and presents them with matches to select from.

In this section, the term "form page" refers to this parent page that the Type Chooser is called from. For example, a Create New Part page might require the user to specify a type for a part, as shown below. The user accesses the Type Chooser by clicking the Browse (...) button next to the Type field.

Step 1 of 2: Create New Part

Fields in red italics are required

Part Number or	<input type="text"/>	<input type="checkbox"/> Autoname
Autoname Series	Not Selected	<input type="button" value="..."/>
Type	Radiographic Part	<input type="button" value="..."/>
Custom Revision Level	3	

The Type Chooser opens with the top-level types that were configured to display. Users can click the + signs or uncheck Top Level Only to see sub-types of the displayed top-level types. For instructions on how to use the page, see the AEF Online Help. For instructions on defining which types show in the list, see the *Live Collaboration Administrator's Guide*.

Select Type

begins with * Top Level Only

Types
 Part

By passing parameters to the JSP, the page can be configured to:

- display or omit specific top-level types
- let users choose only one type or multiple types
- display or omit hidden types
- allow users to select abstract types or not
- show icons for types or not

- call a reload method on the form page to updated values based on the selected type

Internationalized Selected Type Names

The Type Chooser supports internationalization. The system looks up the values for types selected from the chooser in the emxFrameworkStringResource.properties file.

The value(s) is(are) stored in the hidden field on the form, as defined in fieldNameActual. For example, if a user who has their browser set to French selects the type "Part" from the chooser, the fieldNameDisplay parameter is populated with the internationalized string value, the equivalent of "Part" in French. The Type name is populated in the hidden field passed in with the fieldNameActual parameter.

For details on internationalizing dynamic UI components, see [Internationalizing Dynamic UI Components](#).

About the Page (Configurable or JSP) that Uses a Type Chooser

The Type Chooser can be called from a field defined on a configurable Form page or from any standard JSP.

Either way, the page that calls the Type Chooser must have these two fields defined:

- a hidden field that stores the actual type name as stored in the ENOVIA Live Collaboration database.
This is the field defined in the fieldNameActual parameter passed to emxTypeChooser.jsp.
- a field that displays the type(s) selected in the chooser.
This is the field defined in the fieldNameDisplay parameter.

The configurable Form page creates these two fields automatically when you add a field to the web form administrative object that has a RangeHref defined. The hidden field has the same name as the RangeHref field added to the form. The display field has the same name with Display appended to the end of the field name. To call the Type Chooser, the URL entered for the RangeHref must be emxTypeChooser.jsp and its associated parameters. For details about configuring Form pages and using RangeHref fields, see [User Access to UI Components](#).

When calling the Type Chooser from a standard JSP, you must define a form on the page that contains a text field to display the selected type and a hidden field. For example:

```
<form name="searchPage" method="post" action="somejsp.jsp">
<input type="text" name="txtType" value="" 
onClick="showTypeChooser()">
..
<input type="hidden" name="txtSelectedTypeName" value="">
</form>
```

The URL to call the configurable Type Chooser would look like this:

```
../common/
emxTypeChooser.jsp?fieldNameDisplay=txtType&fieldNameActual=txt
SelectedTypeName&formName=searchPage&SelectType=multiselect&Sel
ectAbstractTypes=true&
InclusionList=eServiceEngineeringCentral.Types&observeHidden=tr
ue&SuiteKey=eServiceSuiteEngineeringCentral>ShowIcons=true
```

If you add a new type in ENOVIA Live Collaboration (and have registered it) and want to include it in the Type Chooser,

reload the cache using  > Utilities > Reload Cache.

Parameters Used by the Type Chooser

The emxTypeChooser.jsp page accepts the parameters listed in this table.

Parameter	Description	Accepted Input Values
*fieldNameActual	<p>Use to specify the field on the form page to populate the type(s) the user selects in the Type Chooser. The system populates the specified field with the actual Type names, as stored in the database, so the field must be a hidden field. When calling the Type Chooser using a RangeHref on a configurable Form page, the hidden field is created automatically and has the same name as the RangeHref field.</p> <p>Other pages that need to get the type name selected by the user, for example to perform a search, should get the type name from this field instead of the fieldNameDisplay.</p>	<p>The name of a hidden field on the form page. For example, if the form page contains a field defined as follows:</p> <pre><input type="hidden" name="txtSelectedTypeName" value=""></pre> <p>This parameter should be passed to emxTypeChooser.jsp:</p> <pre>fieldNameActual=txtSelectedTypeName</pre>
*fieldNameDisplay	<p>Use to specify the text field on the form page to populate the type(s) the user selected in the Type Chooser and display these types to the user. When calling the Type Chooser using a RangeHref on a configurable Form page, the display field is created automatically and has the same name as the RangeHref field with "Display" appended to it.</p> <p>This field contains the internationalized version of the selected type list.</p>	<p>The name of a non-hidden field on the form page. For example, if the form page contains a field defined as follows:</p> <pre><input type="text" name="txtType" value="" onClick="showTypeChooser()"></pre> <p>This parameter should be passed to emxTypeChooser.jsp:</p> <pre>fieldNameDisplay=txtType</pre>
ExclusionList	<p>Use to define the types to exclude from the top-level list of types that display when the Type Chooser opens.</p> <p>The inclusion list and the exclusion list cannot both be passed as parameters. If they are, an error message is displayed. If neither is specified, all types are listed. The chooser displays all types in the top level of the hierarchy, even when Top Level Only is unchecked.</p>	Same as InclusionList.
formName	Use to specify the name	The name of a form on the form page. For example, if the form

	<p>of the form that holds the field specified in <code>fieldNameDisplay</code>. If this parameter is not passed in, the Type Chooser looks for the field name on the first form of the form page. If the field name is not found in the first form, then an error message will be presented.</p>	<p>page contains a form defined as follows:</p> <pre><form name="searchPage" method="post" action="somejsp.jsp"></pre> <p>This parameter should be passed to <code>emxTypeChooser.jsp</code>:</p> <pre>formName=searchPage</pre>
<code>frameName</code>	<p>Use to specify the name of the frame that contains the form on the form page. Needed for pages that contain forms in multiple frames.</p>	<p>The name of a frame on the form page.</p>
<code>InclusionList</code>	<p>Use to define the top-level list of types to display when the Type Chooser opens. Note that these types do not have to be top-level types (types with no parents), they will just be listed in the top level in the chooser.</p> <p>The inclusion list and the exclusion list cannot both be passed as parameters. If they are, an error message is displayed. If neither is specified, all types are listed. The chooser displays all types in the top level of the hierarchy, even when Top Level Only is unchecked.</p>	<p>a properties file key</p> <p>If the <code>SuiteKey</code> parameter is passed in, the system looks for the key in the application-specific properties file (for example, <code>emxEngineeringCentral.properties</code>). If the property is not application specific, the system looks in <code>emxSystem.properties</code>. For example, if <code>emxEngineeringCentral.properties</code> contains this property:</p> <pre>eServiceEngineeringCentral.Types=type_Part,type_ECR,type_Sketch</pre> <p>The parameter to pass would be</p> <pre>InclusionList=eServiceEngineeringCentral.Types</pre> <p>a comma delimited list of symbolic names for the types to include:</p> <pre>InclusionList=type_Part,type_ECR</pre>
<code>ObserveHidden</code>	<p>Determines whether or not to display hidden types.</p>	<p>true (default)--Hidden types are not included in the chooser's list of types.</p> <p>false--Hidden types are included.</p>
<code>ReloadOpener</code>	<p>Determines whether to call a <code>reload()</code> method on the opener/parent page (the form page from which the chooser is launched). Reloading the page updates other fields on the form page based on the type(s) selected from the Type Chooser. The method should be a JavaScript function stored in the header section of the JSP or HTML page.</p>	<p>true--The reload method is called.</p> <p>false (default)--The reload method is not called.</p>
<code>SelectAbstractTypes</code>	<p>Determines whether users can select abstract types.</p>	<p>true--Abstract types can be selected.</p> <p>false (default)--Abstract types cannot be selected but they are displayed in the hierarchical list of types.</p>
<code>SelectType</code>	<p>Determines whether the Type Chooser has single select radio buttons or multi-select check boxes.</p>	<p>multiselect--Users can choose multiple types using check boxes.</p> <p>singleselect (default)--Users can choose only one type using a radio button.</p>
<code>ShowIcons</code>	<p>Determines whether to display types icons in the Type Chooser.</p>	<p>true (default)--Type icons are displayed.</p> <p>false--Type icons are not displayed. This improves performance.</p>
<code>SuiteKey</code>	<p>Determines which application's properties file</p>	<p>Possible values for the <code>SuiteKey</code> parameter are: <code>eServiceSuiteTeamCentral</code>, <code>eServiceSuiteEngineeringCentral</code>,</p>

	to look in for the key specified in the InclusionList or ExclusionList parameter. If the SuiteKey is not passed in, emxSystem.properties is used.	eServiceSuiteProgramCentral, or any other suite name.
--	---	---

*Required Setting

Calling the Configurable Type Chooser from a Page

The Type Chooser can be called from a field defined on a configurable Form page or from any standard JSP.

1. To use a property to define a list of types to include or exclude, add the property to the appropriate properties file. When defining the URL for emxTypeChooser.jsp, enter the property key in the InclusionList or ExclusionList parameter. If the file is an application-specific file, you need to specify the application using the SuiteKey parameter.
2. In Business Modeler, open for editing the web form object that contains the field.
3. Open for editing the field that should call the Type Chooser.
4. In the RangeHref parameter for the field, enter the emxTypeChooser.jsp, plus the needed parameters (described in [Parameters Used by the Type Chooser](#)) and a path indication.

The value for the fieldNameActual parameter should be the name of the field you are editing. So if the field's name is PartType, the value should be PartType.

The value for the fieldNameDisplay parameter should be the name of the field you are editing appended with "Display". For a field named PartType, the value for this parameter should be PartTypeDisplay.

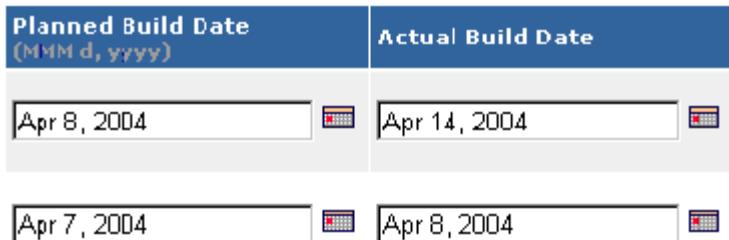
Here are some example Hrefs:

```
 ${COMMON_DIR}/  
emxTypeChooser.jsp?fieldNameActual=PartType&fieldNameDisplay=Pa  
rtTypeDisplay&formName=PartList&SelectType=multiselect&Inclusio  
nList=eServiceEngineeringCentral.Types&ObserveHidden=False&Suit  
eKey=eServiceSuiteEngineeringCentral>ShowIcons=False  
  
 ${COMMON_DIR}/  
emxTypeChooser.jsp?fieldNameActual=PartType&fieldNameDisplay=Pa  
rtTypeDisplay&formName=PartList&SelectType=multiselect&Exclusio  
nList=eServiceEngineeringCentral.Types&ObserveHidden=True&Suite  
Key=eServiceSuiteEngineeringCentral>ShowIcons=True
```

Date Chooser

Column values can be configured to display dates and to let users change the date using a calendar chooser.

The value must be a valid date string. The date value is displayed based on the system and browser locale setting using the "IzDate" taglib. For example:



The format of the value displayed in the column should match the format listed in parenthesis following the date control.

- The first column, Planned Build Date, uses the following settings:

format = date

Editable = true

Allow Manual Edit = true

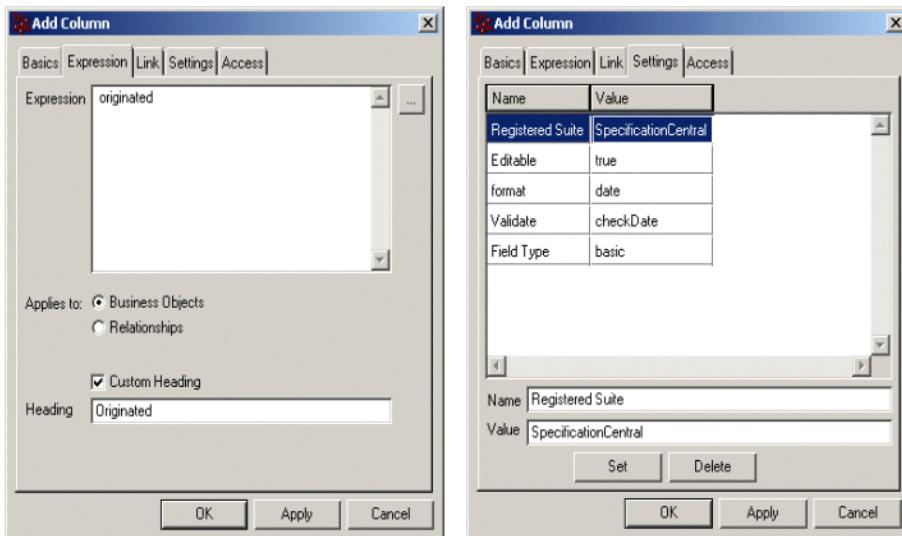
The column header shows the date format depending on the browser locale setting.

- The second column, Actual Build Date, uses the following settings:

format = date

Editable = true

For example, the following graphics show a column set up to display the originated date for the current object. The column is also set to validate the data.



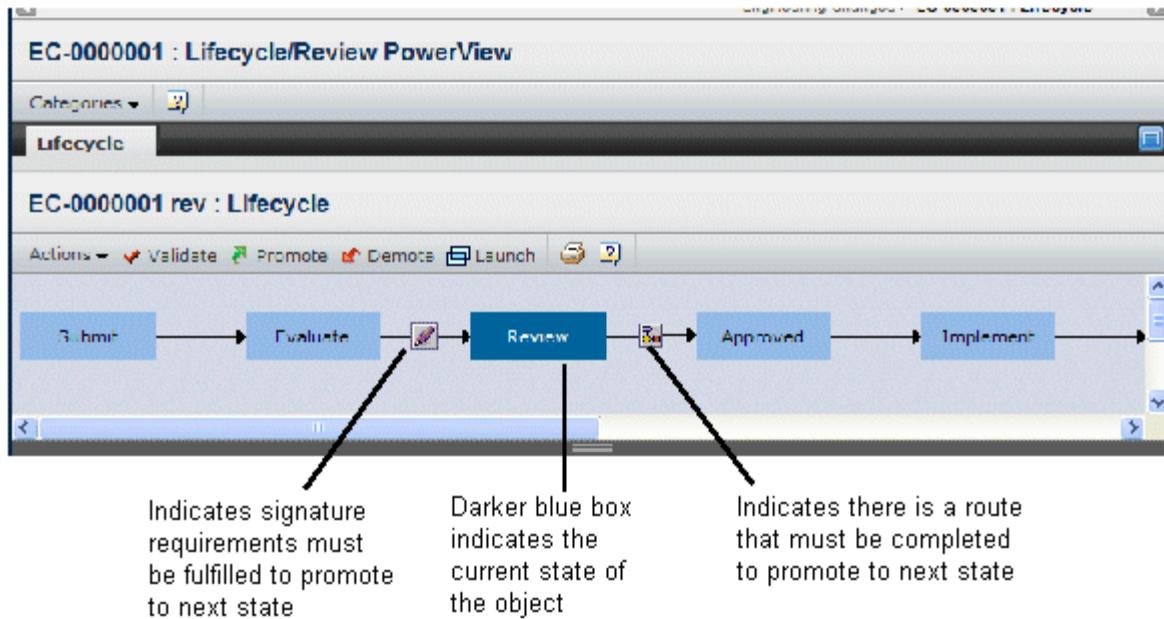
Lifecycle Page

The framework's Lifecycle page, emxLifecycle.jsp, displays the lifecycle for the current business object.

The Lifecycle page includes:

- signature requirements
- branches (does not indicate the branch taken to arrive at a state)
- any blocking routes
- the current state

Users can use the page to view and apply signatures, promote and demote the object, and view blocking routes.



Business Process Services installs a command object called AEFLifecycle that calls the lifecycle page and is configured to be a category in a navigation tree.

You can pass these parameters to emxLifecycle.jsp.

Parameter	Description	Accepted Input Values
actionMenuName	The name of the menu	Name of any menu object that represents a menu. The framework installs with a command object called AEFLifecycleMenu that includes a Promote and Demote link.
header	The title for the page.	Text string or string resource, such as: Lifecycle emxFramework.Lifecycle.LifeCyclePageHeading

*Required Setting

When configured as part of a navigation tree for an object (implemented in the UI as the Categories menu), the system automatically passes the objectID for the object. From the object ID, the page gets the object's type and policy. From the policy, it retrieves the states, signatures, and branches. For policies with branches, when a user clicks Promote, the system promotes to the state for which signature requirements have been met. When naming signatures for branched states, do not use the prefix "GoTo" or the signature box on the Lifecycle page will be hidden, preventing users from being able to apply signatures.

Logout Command

Business Process Services installs a command object, called `AEFLogoutToolbar`, that represents the Logout tool.

This command is assigned to the `AEFGlobalToolbar` menu. When a user clicks the Logout tool, the system calls `emxLogout.jsp`, which ends the current session and returns to the Login page.

If in a Single Signon (SSO) environment, the SSO server should be configured to bypass `emxLogout.jsp`. The user must exit the browser to log out. For instructions on how to bypass `emxLogout.jsp`, see "Single Signon" in the *Live Collaboration Administrator's Guide*.

The tables below describe the parameters and settings used to implement the Logout command. For details on configuring global toolbar tools, see [Menus and Toolbars](#).

Parameter	Description	Installed Input Values
href	The JSP that implements the Logout command. When the JSP is prefixed with <code> \${ROOT_DIR}</code> , the system looks for the JSP in the ematrix root directory, which it gets from <code>emxSystem.properties</code> . The property used is "eServiceSuiteFramework.RootDirectory = .."	<code> \${ROOT_DIR}/emxLogout.jsp</code>
Alt	The internationalization string ID to display as mouse over text.	ID from string resource properties file.
Access	Defines who can access the tool.	All
Setting	Description	Installed Input Values
*Registered Suite	The application the command belongs to. The system looks for image and JSP files in the directory associated with the application. Internationalization details are also obtained based on the Registered Suite.	Framework
Target Location	The frame or window the page should display in.	hiddenFrame
Image	The image displayed on the tool.	buttonToolbarLogout.gif

*Required Setting

Change Password Command

Business Process Services installs a command object, called `AEFChangePasswordToolbar`, that represents the Change Password tool.

This command is assigned to the Tools menu, which is assigned to the My Tools menu, which is assigned to the `AEGlobalToolbar` menu. The Change Password command calls `emxChangePassword.jsp`, which lets users change their own password. The page displays in a popup window. Before making the password change, the page checks to make sure the current password the user enters is correct and makes sure the new password and verify new password entries are the same.

The tables below describe the parameters and settings used to implement the Change Password command. For details on configuring global toolbar tools, see [Global Toolbar Menu](#).

Parameter/Setting	Description	Installed Input Values
href	The JSP that implements the Change Password page.	<code>emxChangePassword.jsp</code>
Alt	The internationalization string ID to display as mouse over text.	<code>emxNavigator.UIMenu.ChangePassword</code>
Access	Defines who can access the tool.	All
Setting	Description	Installed Input Values
Help Marker	Specifies the name of the help marker to call for context-sensitive help. For information about implementing help, see Implementing Context-Sensitive Help for FrameMaker Source .	String The naming convention for help markers is "emxhelp" followed by the object or feature and then the action, for example, <code>emxhelproutecreate</code> and <code>emxhelpprojectedit</code> . The marker is all lowercase with no spaces.
*Registered Suite	The application the command belongs to. The system looks for image and JSP files in the directory associated with the application. Internationalization details are also obtained based on the Registered Suite.	Framework
Target Location	The frame or window the page should display in. By default, the page displays in a slidein frame. The target location can be changed to any specific existing frame name to display it in that frame.	slidein The window is modal.
Image	The image displayed on the tool.	<code>buttonToolbarPassword.gif</code>

*Required Setting

Page History Page and Command

Business Process Services installs a command object called AEFPAGEHistoryToolbar that represents the Page History menu command. Page History is a list of visited pages and is different from the History page (which lists actions performed on an object).

This command is assigned to the Tools menu, which is assigned to the My Tools menu, which is assigned to the AEFGlobalToolbar menu. The Page History command calls emxPageHistory.jsp, which lists the last 50 pages the user has visited since the user logged in. The number of pages stored by the page is configurable using the `emxSystem.pageHistory.limit=50` property in emxSystem.properties. In addition, the `emxFramework.History.maxRecords=2000` property defines how many pages will be retained for all users. Users can use the page to revisit the pages, bookmark them, and copy and paste the URLs for the pages into emails to invite other users to visit the pages. The page displays in a non-modal popup window.

For information about how to use the page, click the Help button on the page in the application.

Page History

[Refresh](#) [?](#)

Link Name	Suite/Category	Address (URL)
Workspace 100		http://hostname/ematri
Workspaces	Team	http://hostname/ematri
Search	Sourcing	http://hostname/ematri
Comp1 Package 1	Attachment	http://hostname/ematri
Comp1 Package 1		http://hostname/ematri
Packages	Sourcing	http://hostname/ematri
Comp1 RFQ 17.1	Supplier	http://hostname/ematri
Comp1 RFQ 17.1	Line Item	http://hostname/ematri
Comp1 RFQ 17.1		http://hostname/ematri
RFQ's	Sourcing	http://hostname/ematri

To create a Favorite or Bookmark:

Internet Explorer (Windows)  Right-click a link and select "Add to Favorites".

Netscape (Windows)  Right-click a link and select "Add Bookmark".

Other platforms
The procedures will be similar but the mouse-click actions may vary.

[Close](#)

The tables below describe the parameters and settings used to implement the Page History command. For details on configuring menus, see [Menus and Toolbars](#).

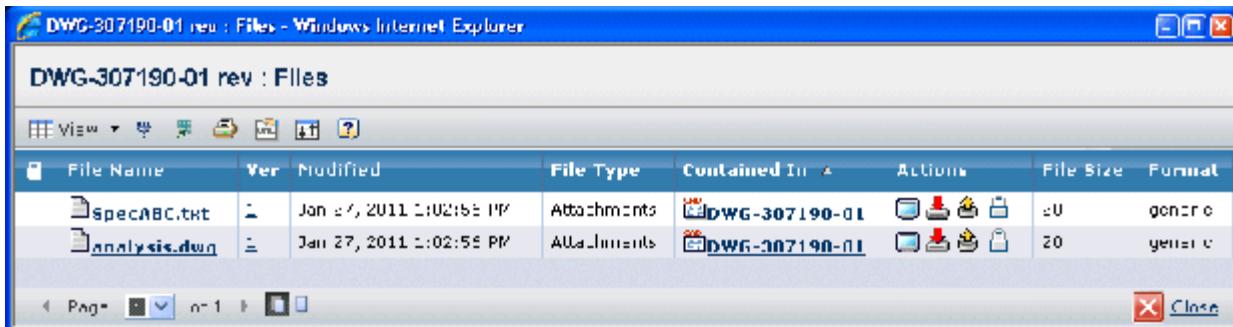
Parameter	Description	Installed Input Values
href	The JSP that implements the Page History page.	emxPageHistory.jsp
Alt	The internationalization string ID to display as mouse over text.	emxFramework.PageHistory.PageHistoryLabel
Access	Defines who can access the tool.	All
Setting	Description	Installed Input Values
Help Marker	Specifies the name of the help marker to call for context-sensitive help. For information about implementing help, see Implementing Context-Sensitive Help for FrameMaker Source .	emxhelppagehistory
*Registered Suite	The application the command belongs to. The system looks for image and JSP files in the directory associated with the application. Internationalization details are also obtained based on the Registered Suite.	Framework

Target Location	The frame or window the page should display in. By default, the page displays in a separate popup window.	popup
Popup Modal	The page is non-modal so the page remains open while users continue to visit pages.	false
Window Height	<p>Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties.</p> <p>The window height for the new popup window. This value is used only when the setting for Target Location is set to "popup".</p>	550
Window Width	<p>Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties.</p> <p>The window width for the new popup window. This value is used only when the setting for Target Location is set to "popup".</p>	550
Image	The image displayed in the menu.	ButtonToolbarPageHistory.gif

*Required Setting

Quick File Access Details Table

When a table contains a column of type File, the hyperlinked paper clip icon opens a Quick File Access details page.



The page toolbar contains the standard filter, export, printer-friendly, and help commands. The table includes these columns:

Column Name	Description
	Lock status. If empty, the file is unlocked, if the lock displays, the file is locked.
File Name	Name of the file connected to the selected object.
Ver	File Version Identifier.
Modified	The date and time of file creation.
File Type	The type of file.
Contained In	The Title of the document (if available), or the context object of the checked in file.
Actions	See Common Components User's Guide
File Size	The size of the file.
Format	The format of the file

Context-Sensitive Help

ENOVIA applications include context-sensitive help; and you can provide context-sensitive help for any custom applications.

In this section:

- [About Context-Sensitive Help](#)
- [Implementing Context-Sensitive Help for FrameMaker Source](#)
- [Implementing Context-Sensitive Help for XML Source](#)

About Context-Sensitive Help

Context-sensitive help is provided for all ENOVIA products and is accessed by the user clicking the ? icon on a toolbar.

The help systems for ENOVIA products are written either as FrameMaker files converted to HTM files, or as XML file converted to HTM files. While the content of the help systems are independent of the source document format, the method of accessing the help system from the products is different.

The following topics are discussed:

- [Context-Sensitive Help \(FrameMaker Source\)](#)
- [Context-Sensitive Help \(XML Source\)](#)
- [Lifecycle Helpmarkers](#)

Context-Sensitive Help (FrameMaker Source)

Each ENOVIA product has its own help system. All help systems are installed in ENOVIA_INSTALL/STAGING/ematrix/doc/SUITE_NAME/LANG, where SUITE_NAME is the "registered suite" name of the application.

For example, the English help for Engineering Central could be in .../ematrix/doc/engineeringcentral/en.

When the application is deployed (with warutil) the English documentation is put in WEBROOT/WARAPPNAME/doc/SUITE_NAME/en. Each ENOVIA product web page includes a help button which calls the wwhelp.htm file within its registered suite directory. The same file name is called for all application help systems, each within its own registered suite directory.

When you call the wwhelp.htm file, the help system opens to its title page. To call a particular topic within the help system, you must pass the help marker for that topic. (Placing help markers into the help system is part of building the help files and beyond the scope of this document. To get the help marker that should be called for an application page, consult the person responsible for creating the help files.)

If the system cannot find WEBROOT/WARAPPNAME/doc/SUITE_NAME/en/wwhelp.htm, it displays an error message.



Context-Sensitive Help (XML Source)

The help built from XML source files calls the needed help page using a lookup table defined in emxSystem.properties for the application.

The emxSystem.properties file contains these properties for each product that has a help system build from XML source files:

```
eServiceSuiteSUITE NAME.Help.Directory = XXXUserMap  
eServiceSuiteSUITE NAME.HelpFile = emxXXXHelp.properties
```

where:

- **SUITE NAME** is the name of the ENOVIA product, such as **EngineeringCentral** or **Requirements**.
- **XXX** is the product identifier, such as ENG for Engineering Central or RMT for ENOVIA Requirements Central

For example:

```
eServiceSuiteEngineeringCentral.Help.Directory = EngUserMap  
eServiceSuiteEngineeringCentral.HelpFile = emxENGHelp.properties
```

The Help.Directory (XXXUserMap) contains these items:

- HTM files
- Sitemap.htm (table of contents)
- Index.htm (index for the help system)

You can use the files in an existing product's help system as a guideline for how to create a Sitemap and Index for a custom help system.

The HelpFile contains the lookup table that maps helpmarkers used on the product page to the specific page within the help system for help about that page.

The lookup table is contained in a file named **emxXXXHelp.properties**, where **XXX** is the product indicator. For example, **emxENGHelp.properties** is the lookup file for Engineering Central. The entries in this file use this format:

```
helpmarker = HTM File Name
```

For example:

```
emxhelpsearchservice = bps-t-webservices-search.htm
```

The lookup file must contain all the helpmarker used in the product.

This design requires that all helpmarkers for all products be unique. For example, if Engineering Central has a helpmarker named **emxhelpsearch**, then no other application can use that helpmarker. For custom applications, you may want to include a prefix to make sure you do not inadvertently re-use any helpmarkers currently used by an ENOVIA product, such as

`customhelpsearch`.

The `emxXXXHelp.properties` file must be installed in the properties folder for that application, such as `EBOUNT/CNext/Properties`.



Lifecycle Helpmarkers

Lifecycle help markers are called based on the name of the policy that controls the object's lifecycle.

The convention for naming lifecycle help markers is "emxhelppolicy" followed by an underscore and then the policy name. For example, the policy that controls most parts is the EC Part policy. The help marker is `emxhelppolicy_ecpart`.

Implementing Context-Sensitive Help for FrameMaker Source

Use the following steps to call a particular help topic for a configurable page when using a help system developed in FrameMaker.

Before you begin: This procedure assumes the help system has the directory structure and `wwhelp.htm` file described above and that the corresponding help marker has been included in the help system.

1. Make sure the command object that calls the configurable page, such as `emxTables.jsp` or `emxFoms.jsp`, has the setting:

`Registered Suite=APPLICATION`

Where `APPLICATION` is the same as the suite name defined in the key `eServiceSuites.DisplayedSuites` within `emxSystem.properties`. The "eServiceSuite" prefix, if any, can be left off.

Note that even if there is no help marker defined, as described in the next step, when the configurable page's Help button is clicked, the appropriate application help system is still called based on the Registered Suite. The help system opens to the title page.

2. Pass the help marker to the configurable page in either of these ways:

- In the command object that calls the configurable page, add the setting:

`Help Marker=MARKER`

Where `MARKER` is the help marker for the help topic to call, for example, `emxhelpoutecreate`. The command will automatically append the `HelpMarker` parameter to the URL for the configurable page.

- In the href parameter for the command object that calls the configurable page, manually enter the parameter:

`HelpMarker=MARKER`

This example shows a URL for a configurable table that includes the help marker:

```
 ${COMMON_DIR}/  
 emxTable.jsp?inquiry=PartList,PTParts,CMParts&inquiryLabel=emxE  
 ngineeringCentral.Common.All,PT Parts,CM  
 Parts&table=DemoPartTable&header=emxEngineeringCentral.Common.P  
 arts&toolbar=DemoPartToolbar&TipPage=emxBlank.jsp&sortColumnName  
 e=Name&FilterFramePage2=${COMMON_DIR}/  
 emxTableFilterIncludeSample.jsp&FilterFrameSize=40&selection=mu  
 ltiple&SubmitURL=${SUITE_DIR}/  
 emxBlank.jsp&SubmitLabel=emxEngineeringCentral.Button.Next&Canc  
 elButton=true&CancelLabel=emxEngineeringCentral.Common.Cancel&  
 ubHeader=Bill of Material Level  
 1&pagination=8&headerRepeat=12&rememberSelection=true&HelpMarke  
 r=emxhelppartlist
```

Implementing Context-Sensitive Help for XML Source

Use the following steps to call a particular help topic for a configurable page when using a help system developed in XML. The help systems for ENOVIA products are converted to HTM files from XML source files. You can develop your HTM files using any appropriate development tool.

1. Make sure the command object that calls the configurable page, such as emxTables.jsp or emxFoms.jsp, has the setting:

Registered Suite=APPLICATION

Where APPLICATION is the same as the suite name defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. The "eServiceSuite" prefix, if any, can be left off.

2. Create properties in emxSystem.properties for your help system:

```
eServiceSuiteSUITENAME.Help.Directory = XXXUserMap  
eServiceSuiteSUITENAME.HelpFile = emxXXXHelp.properties
```

where:

- SUITENAME is the name of the custom product
- XXX is the product identifier

The SUITENAME must match the APPLICATION name used in Step 1.

3. Create the Sitemap.htm, Index.htm, and help *.htm files. To take advantage of the styles and formats used by other ENOVIA products, refer to an existing file and copy the necessary header lines (including the meta tags, customized for the specific file).
4. Create the lookup table. See [Context-Sensitive Help \(XML Source\)](#) for details and put it in the Properties folder for the custom application.
5. Pass the help marker to the configurable page in either of these ways:

- In the command object that calls the configurable page, add the setting:

Help Marker=MARKER

Where MARKER is the help marker for the help topic to call, for example, emxhelpoutcreate. The command will automatically append the HelpMarker parameter to the URL for the configurable page.

- In the href parameter for the command object that calls the configurable page, manually enter the parameter:

HelpMarker=MARKER

This example shows a URL for a configurable table that includes the help marker:

```
 ${COMMON_DIR}/  
emxTable.jsp?inquiry=PartList,PTParts,CMParts&inquiryLabel=emxE  
ngineeringCentral.Common.All,PT Parts,CM  
Parts&table=DemoPartTable&header=emxEngineeringCentral.Common.P  
arts&toolbar=DemoPartToolbar&TipPage=emxBlank.jsp&sortColumnName  
e=Name&FilterFramePage2=${COMMON_DIR}/  
emxTableFilterIncludeSample.jsp&FilterFrameSize=40&selection=mu  
ltiple&SubmitURL=${SUITE_DIR}/  
emxBlank.jsp&SubmitLabel=emxEngineeringCentral.Button.Next&Canc  
elButton=true&CancelLabel=emxEngineeringCentral.Common.Cancel&  
ubHeader=Bill of Material Level  
1&pagination=8&headerRepeat=12&rememberSelection=true&HelpMarke  
r=emxhelppartlist
```

Web Services

A Web service is a Web application that dynamically interacts with other Web applications using open standards that include XML, UDDI and SOAP. You can develop your own web services.

In this section:

- [About Web Services](#)
- [Developing and Deploying a Web Service](#)
- [Web Service Examples](#)

About Web Services

A Web service is a Web application that dynamically interacts with other Web applications using open standards that include XML, UDDI and SOAP. Web services can be written to run in the background to perform tasks such as checking in files, or to provide integrations to other software packages such as word processing or spreadsheet programs.

The following topics are discussed:

- [Web Services](#)
- [Requirements and Set Up](#)
- [Marking the JPO](#)
- [Types of Scoped Services](#)
- [Handling Data](#)
- [Authentication](#)
- [Development and Deployment](#)

Web Services

Web services are ideal for use in large enterprises to exchange data between divisions and subsidiaries or with partners and clients.

In this environment, a service, or rather its client-side interface, is provided to collaboration partners and can easily be added to their applications. The service is registered on a server, allowing the application to search for and find the service and then to seamlessly exchange data with it. Most wireless devices and even some cars available today are Java-enabled and so client-side midlets can be written as services for this type of access.

Web services are enabled within ENOVIA Live Collaboration via an integration to Apache Axis Web service toolkit. For information on Axis, see <http://ws.apache.org/axis>.

When installing ENOVIA Live Collaboration Server or one of the application servers, Web services are enabled by default (that is, the Axis requirements are included). Then, to create a Web service, you can write Java program objects (JPOs) and mark them as a Web service (see [About Web Services](#)). When a JPO is marked as a Web service and compiled (via the compile button in Business Modeler or through MQL), the ENOVIA Live Collaboration:

- generates and compiles service stub classes.
- creates Axis deployment files (.wsdd files)
- deploys the service

In this way enterprises can leverage their existing intellectual property and deploy any existing JPO as a Web service.



Requirements and Set Up

When you install the Studio Modeling Platform or an ENOVIA Live Collaboration Server, Web services are enabled by default and can be configured using environment variables.

The following environment variables are related to creating Web services with ENOVIA Live Collaboration. On your development machine the first variable is required, the second is optional. On the server machine you do not need either variable.

- **MX_SERVICE_PATH** specifies the directory where the service stubs and deployment files will be written when service JPOs are compiled in the ENOVIA Live Collaboration. If MX_SERVICE_PATH points to the directory where the Axis Web application is deployed, the classes will be available immediately to the application server. If another directory is used, the service stubs and deployment files must be moved to the application server through some process outside of the ENOVIA Live Collaboration (that is, copied manually). For example:

MX_SERVICE_PATH=%catalina_home%\webapps\web-inf\classes

- **MX_SERVICE_ADMIN** specifies the administration URL of the Axis Web application. If specified, the ENOVIA Live Collaboration will automatically deploy Web services when programs are compiled (and undeploy when programs are deleted, renamed, etc.). This value should only be set when MX_SERVICE_PATH is pointing to the directory where Axis is running. For example:

MX_SERVICE_ADMIN=http://localhost:8080/WEBAPP_NAME



Marking the JPO

There are various styles of Web services, and you can create different service styles in a JPO.

You mark the JPO as a service by implementing one of the following ENOVIA Live Collaboration Service classes within the JPO code:

- MatrixService

- MatrixDocumentService
- MatrixMessageService
- MatrixWrappedService

Document and wrapped services are common when using XML schema or XML Beans to build a service description. Message services are used when you need to process raw XML messages. The MatrixService interface is used for all other java:rpc style services. See Axis documentation at <http://ws.apache.org/axis/java/user-guide.html> for more information on the various service styles.



Types of Scoped Services

Web services may be scoped as either session or request.

With session scoped services, the object that implements the service is instantiated once per session and reused for each service request. When developing Web services with ENOVIA Live Collaboration, a JPO will be session scoped if it has a constructor with the signature:

```
Public classname(Context context) (any # of args);
```

However, the ENOVIA Live Collaboration classloader requires a constructor with one of the following signatures:

```
Public classname();  
Public classname(Context context, String[] args);
```

Therefore, if you use a constructor that does not match one of these two signatures, you must also define the default constructor if you want the service to be session scoped.

With request scoped services, service objects are instantiated (and destroyed) per request. If the JPO does not meet the constructor requirements described above, then it will be request scoped. In general, request scoping should be used, as it consumes fewer resources on the application server. Request scoped services do not require maintenance of an http session on the application server.

When deploying Web services, specify the methods within a JPO that are visible outside of the ENOVIA Live Collaboration (that is, as part of the service). Currently, when writing any kind of service, all public methods in the JPO are exposed.



Handling Data

Arguments passed to JPOs and values returned by JPOs that are not simple data types (int, double, boolean) require special handling when the JPO is used as a service.

Axis provides mapping for many commonly used Java classes (String, Date, Map, ArrayList); see Axis documentation for a complete listing. Refer to [Returning a String - JPO Code](#) for an example.

Axis also provides mapping for javax.activation.DataHandler, used for building services with attachments. When these standard classes are used, the standard Axis handling is used. Refer to [Checkin with Attachments](#) for an example.

Another JPO may be passed as an argument (or return value) to or from a service JPO. When this programming style is used, ENOVIA Live Collaboration assumes that the JPO argument implements the Value Bean model, with getter/setter methods for each field in the JPO. Refer to [General Query](#) for an example of returning a value bean from a JPO. In summary, the arguments that may be passed to or from service JPOs are:

- simple data types (int, double, boolean, and so on)
- Java classes mapped by Axis (String, Map, Date, ArrayList, and so on)
- javax.activation.DataHandler (attachments)
- JPO Value Beans

Any other argument type will result in errors either during deployment or execution of the service.



Authentication

Clients must authenticate themselves to services that they are calling.

Axis provides a mechanism for passing a username/password in the Web service message context. These credentials are transmitted over the wire in an HTTP basic authentication header. Currently this is the only authentication mechanism supported by ENOVIA Web services.



Development and Deployment

When creating services with ENOVIA Live Collaboration, you should use Java naming conventions for packages and classes as the names of the JPOs. ENOVIA Live Collaboration will then respect those conventions, and build service stubs that reside in a directory hierarchy corresponding to the package names.

Also, the Web service name derives its name from the JPO using Java conventions for capitalization; initial capital for each name part, using a period (.) for the delimiter. For instance, if a JPO named `matrix.axis.myService` is compiled, the following files and directory structure are created:

```
%MX_SERVICE_PATH%\matrix\axis\MatrixAxisMyService.class  
%MX_SERVICE_PATH%\matrix\axis\MatrixAxisMyService.java  
    %MX_SERVICE_PATH%\MatrixAxisMyServiceDeploy.wsdd  
    %MX_SERVICE_PATH%\MatrixAxisMyServiceUndeploy.wsdd  
and, if MX_SERVICE_ADMIN is set:  
%MX_SERVICE_PATH%\MatrixAxisMyService.wsdl
```

You should develop and maintain services as part of logical packages.

Note:

You cannot include double-byte characters in the name of a JPO or compilation as a service will fail.

Also, do not include .java in the name of a JPO, as it will be referred to without the .java in the web service stub file that is created, in an attempt to have the webservice load the JPO by program name.

Use an exploded war file when developing Web services. This allows you to have an application server running and updated automatically with each program compilation. After development is complete, the files can be copied from this development area to the STAGING directory to be packed up using the ENOVIA Live Collaboration War Utility.

As mentioned above, the MX_SERVICE_PATH and MX_SERVICE_ADMIN settings control how Web service stubs are written and registered. In addition, these stubs reference the Java HttpServletRequest class and so it must be available during compilation. This class is provided to each application server via the following .jar files:

- BEA WebLogic 7: BEA_HOME/weblogic700/server/lib/weblogic.jar
- IBM WebSphere 5: WAS_HOME/lib/j2ee.jar
- Apache Tomcat 4: CATALINA_HOME/common/lib/servlet.jar
- Sun Java System Application Server 7: AS_INSTALL/lib/appserv-ext.jar

The appropriate .jar file must be referenced in MX_CLASSPATH or compilation will fail.

While dynamic compilation of Web service JPOs is possible, force compiling them is recommended.

Developing and Deploying a Web Service

You can develop and deploy a Web service with the ENOVIA Live Collaboration.

1. Create or edit a JPO that implements MatrixService (or one of the other supported interfaces).
2. Start the application server with exploded Web application deployed. This Web application can be an empty ENOVIA Web application generated by running the ENOVIA Live Collaboration War Utility against an empty STAGING directory.
3. Compile the JPO with proper ini entries pointing to development Webapp and appserver. For example:

```
MX_SERVICE_PATH=%catalina_home%\webapps\web-inf\classes  
MX_SERVICE_ADMIN=http://localhost:8080/WEBAPP_NAME
```

4. Test the Web service by accessing the URL of the exploded Web application:

```
http://<hostname>:<port>/<webapp>/services/<webservicename>
```

5. If deploying .war/.ear file:

- a. Create a directory called `classes` under `STAGING/ematrix/WEB-INF/`.
- b. Either copy generated files from the development Webapp/WEB-INF/classes or run MQL `compile prog * force update` command with `MX_SERVICE_PATH` set to `STAGING/ematrix/WEB-INF/classes`.
- c. Copy generated deployment file `WEB-INF/server-config.wsdd` to `STAGING/ematrix/WEB-INF`.
- d. Run ENOVIA Live Collaboration War Utility.
- e. Deploy Web Application.
- f. Test Web service by accessing URL:

```
http://<hostname>:<port>/<webapp>/services/<webservicename>
```

If `MX_SERVICE_ADMIN` is set, when you compile it deploys the Web service to that Web application and modifies the `server-config.wsdd` in the WEB-INF directory. If you make modifications to the signature of an exposed method, the application server must be restarted after compilation. For any other modifications to the code, the changes are immediate -- no restarting of the Web server is required.

Web Service Examples

This section provides several examples of writing web services.

- [Returning a String - JPO Code](#)
- [Returning a String - Client-side Code](#)
- [General Query](#)
- [Checkin with Attachments](#)

Returning a String - JPO Code

```
import matrix.db.*;
import matrix.util.*;
import java.io.*;
import java.util.*;
public class ${CLASSNAME} implements MatrixService {
    public String hello(Context ctx, String arg)
    {
        return "Hello " + arg + ", How are you doing ?";
    }
}
```



Returning a String - Client-side Code

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import javax.xml.namespace.QName;
import javax.xml.rpc.ParameterMode;
public class Client
{
    public static void main(String [] args)
    {
        try {
            if ((args == null) || (args.length < 1)) {
                System.out.println("Nothing passed in...");
                System.exit(0);
            }
            String endpointURL = args[0];
            String textToSend;
            if (args.length < 2) {
                textToSend = "<nothing>";
            } else {
                textToSend = args[1];
            }
            Service service = new Service();
            Call call = (Call) service.createCall();
            call.setTargetEndpointAddress( new
java.net.URL(endpointURL) );
            call.setOperationName( new QName("", "hello") );
            call.addParameter( "arg1", XMLType.XSD_STRING,
ParameterMode.IN );
            call.setReturnType(
org.apache.axis.encoding.XMLType.XSD_STRING );
            String ret = (String) call.invoke( new Object[] {
textToSend } );
            System.out.println("String Returned : " + ret);
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```



General Query

The following program implements a query Web service. The program defines a service entry point (i.e. method), which takes some arguments, runs a query using the Studio Customization Toolkit, and packages up the results. The results are returned as an array of Value Beans.

Source for JPO 'matrix.axis.Query':

```
import matrix.db.*;
import matrix.util.*;
import java.util.*;
public class ${CLASSNAME} implements MatrixService {
    public class QueryResult{
        String type, name, revision, owner, current, description;
        public String getType() { return type; }
        public void setType(String s) { type = s; }
        public String getName() { return name; }
        public void setName(String s) { name = s; }
        public String getRevision() { return revision; }
        public void setRevision(String s) { revision = s; }
        public String getOwner() { return owner; }
        public void setOwner(String s) { owner = s; }
        public String getCurrent() { return current; }
        public void setCurrent(String s) { current = s; }
        public String getDescription() { return description; }
        public void setDescription(String s) { description = s; }
    }
    public QueryResult[] evaluate(Context context, String type,
String name, String revision, String owner, String vault, String
where) throws Exception {
        Query query = new Query("");
        query.setBusinessObjectType(type);
        query.setBusinessObjectName(name);
        query.setBusinessObjectRevision(revision);
        query.setOwnerPattern(owner);
        query.setVaultPattern(vault);
        query.setWhereExpression(where);
        StringList select = new StringList();
        select.add("type");
        select.add("name");
        select.add("revision");
        select.add("owner");
        select.add("current");
        select.add("description");
        BusinessObjectWithSelectList list = query.select(context,
select);
        QueryResult[] result = new QueryResult[list.size()];
        for (int i=0; i<list.size(); i++) {
            BusinessObjectWithSelect o = list.getElement(i);
            QueryResult r = new QueryResult();
            r.setType(o.getSelectData("type"));
            r.setName(o.getSelectData("name"));
            r.setRevision(o.getSelectData("revision"));
            r.setOwner(o.getSelectData("owner"));
            r.setCurrent(o.getSelectData("current"));
            r.setDescription(o.getSelectData("description"));
            result[i] = r;
        }
        return result;
    }
}
```

PocketSoap client

The following is a Microsoft Excel macro that invokes the query service using pocket soap. The macro is invoked from a form that specifies the parameters for the query. The results are loaded into the excel spreadsheet:

```
Private Sub CommandButton1_Click()
    Dim env
    Set env = CreateObject("pocketSOAP.Envelope.2")
```

```

Const scfScripting = 1
env.SerializerFactory.SetConfig (scfScripting)

env.SetMethod "evaluate", "urn:MatrixService"
env.Parameters.Create "type", txtType.Value
env.Parameters.Create "name", txtName.Value
env.Parameters.Create "revision", txtRevision.Value
env.Parameters.Create "owner", txtOwner.Value
env.Parameters.Create "vault", txtVault.Value
env.Parameters.Create "where", txtWhere.Value

Dim http
Set http = CreateObject("pocketSOAP.HTTPTransport.2")
http.Authentication "creator", ""
http.SOAPAction = ""
http.Send "http://localhost:8080/axis/services/
MatrixAxisQuery", env.serialize
env.Parse http
Dim arrRes
arrRes = env.Parameters.Item(0).Value
Cells.Clear
For idx = LBound(arrRes) To UBound(arrRes)
    Cells(idx + 1, 1) =
arrRes(idx).Nodes.ItemByName("type").Value
    Cells(idx + 1, 2) =
arrRes(idx).Nodes.ItemByName("name").Value
    Cells(idx + 1, 3) =
arrRes(idx).Nodes.ItemByName("revision").Value
    Cells(idx + 1, 4) =
arrRes(idx).Nodes.ItemByName("owner").Value
    Cells(idx + 1, 5) =
arrRes(idx).Nodes.ItemByName("current").Value
    Cells(idx + 1, 6) =
arrRes(idx).Nodes.ItemByName("description").Value
Next
Range("A1").Select
Me.Hide
End Sub

```



Checkin with Attachments

JPO code for Web Service to handle a checkin and checkout:

```

import matrix.db.*;
import matrix.util.*;
import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import java.io.File;
public class ${CLASSNAME} implements MatrixService{
    String sep = System.getProperty("file.separator");
    public void checkin(Context context, String type, String name, String rev, String
    vault, String format, String store, String file, boolean append, boolean unlock,
    DataHandler data) throws Exception {
        context.connect();
        File f = new File(data.getName());
        f.renameTo(new File(f.getParent() + sep + file));
        BusinessObject bo = new BusinessObject(type, name, rev, vault);
        bo.open(context);
        bo.checkinFile(context, unlock, append, "", format, file, f.getParent());
        bo.close(context);
    }
    public DataHandler checkout(Context context, String type, String name, String rev,
    String vault, String format, String file, boolean lock) throws Exception {
        context.connect();
        BusinessObject bo = new BusinessObject(type, name, rev, vault);
        bo.open(context);
        bo.checkoutFile(context, lock, format, file, context.createWorkspace() + sep);
        bo.close(context);
    }
}

```

```

        return (new DataHandler(new FileDataSource(context.createWorkspace() + sep +
file)));
    }
}
Client-side code for Checkin with Web Service:
import org.apache.axis.AxisFault;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.encoding.ser.JAFDataHandlerDeserializerFactory;
import org.apache.axis.encoding.ser.JAFDataHandlerSerializerFactory;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.xml.namespace.QName;
import javax.xml.rpc.ParameterMode;
import java.util.*;
import java.io.*;
public class ClientCheckin {
    public static void main(String[] args) throws Exception {
        Service service = new Service();
        Call call = (Call) service.createCall();
        call.setTargetEndpointAddress(args[0]);
        // *****
        // File checkin
        // *****
        // Set the arguments for the call
        call.setOperationName(new QName("urn:MatrixService", "checkin"));
        call.setUsername("creator");
        QName fileAttachment = new QName("urn:MatrixService", "DataHandler");
        call.registerTypeMapping(DataHandler.class,
            fileAttachment,
            JAFDataHandlerSerializerFactory.class,
            JAFDataHandlerDeserializerFactory.class);
        call.addParameter("type", XMLType.XSD_STRING, ParameterMode.IN);
        call.addParameter("name", XMLType.XSD_STRING, ParameterMode.IN);
        call.addParameter("rev", XMLType.XSD_STRING, ParameterMode.IN);
        call.addParameter("vault", XMLType.XSD_STRING, ParameterMode.IN);
        call.addParameter("format", XMLType.XSD_STRING, ParameterMode.IN);
        call.addParameter("store", XMLType.XSD_STRING, ParameterMode.IN);
        call.addParameter("file", XMLType.XSD_STRING, ParameterMode.IN);
        call.addParameter("append", XMLType.XSD_BOOLEAN, ParameterMode.IN);
        call.addParameter("unlock", XMLType.XSD_BOOLEAN, ParameterMode.IN);
        call.addParameter("data", fileAttachment, ParameterMode.IN);
        call.setReturnType(XMLType.AXIS_VOID);
        // Invoke the service
        Object[] checkinParams = new Object[10];
        checkinParams[0] = args[1];
        checkinParams[1] = args[2];
        checkinParams[2] = args[3];
        checkinParams[3] = args[4];
        checkinParams[4] = "generic";
        checkinParams[5] = "STORE";
        checkinParams[6] = args[5];
        checkinParams[7] = new Boolean(true);
        checkinParams[8] = new Boolean(false);
        checkinParams[9] = new DataHandler(new FileDataSource(args[5]));
        try {
            Object ret = call.invoke(checkinParams);
        } catch (Exception e) {
            System.err.println("Unable to checkin file because of the following exception:"
+
e.toString());
        }
        // *****
        // File checkout
    }
}
```

```
// ****
call.removeAllParameters();
call.setOperationName(new QName("urn:MatrixService", "checkout"));
call.setUsername("creator");
// Set the arguments for the call
call.addParameter("type", XMLType.XSD_STRING, ParameterMode.IN);
call.addParameter("name", XMLType.XSD_STRING, ParameterMode.IN);
call.addParameter("rev", XMLType.XSD_STRING, ParameterMode.IN);
call.addParameter("vault", XMLType.XSD_STRING, ParameterMode.IN);
call.addParameter("format", XMLType.XSD_STRING, ParameterMode.IN);
call.addParameter("file", XMLType.XSD_STRING, ParameterMode.IN);
call.addParameter("lock", XMLType.XSD_BOOLEAN, ParameterMode.IN);
call.setReturnType( fileAttachment );
// Invoke the service
Object[] checkoutParams = new Object[7];
checkoutParams[0] = args[1];
checkoutParams[1] = args[2];
checkoutParams[2] = args[3];
checkoutParams[3] = args[4];
checkoutParams[4] = "generic";
checkoutParams[5] = args[5];
checkoutParams[6] = new Boolean(false);
DataHandler dh = null;
try {
    dh = (DataHandler)call.invoke(checkoutParams);
    // Write file to local storage
    String sep = System.getProperty("file.separator");
    FileOutputStream os = new FileOutputStream(new File("c:" + sep + args[5]));
    dh.writeTo(os);
    os.close();
} catch (Exception e) {
    System.err.println("Unable to checkout file because of the following exception:"
+
                e.toString());
}
}
```

Triggers

You can use event triggers to implement business rules. Event triggers provide a powerful way to customize ENOVIA product behavior through Program objects.

In this section:

- [About Event Triggers](#)
- [How Rules are Automated](#)
- [Trigger Scenarios](#)
- [Designing Triggers](#)
- [Trigger Programs](#)
- [Adding a Trigger for a Trigger Event](#)
- [Validate Query Trigger](#)
- [Recursion Detection Modes and Limits](#)
- [Enabling/Disabling Triggers](#)
- [Utility Trigger Programs Reference](#)

About Event Triggers

You can use event triggers to implement business rules.

Also see the *Live Collaboration Administrator's Guide* for details on how triggers are used and defined, and a trigger reference. The Administrator's Guide for each ENOVIA product includes a list of triggers used by that product.

The following topics are discussed:

- [Introduction to Triggers](#)
- [User/Password Information for a Trigger](#)
- [User Interaction](#)
- [Lifecycle Checks and Actions with Event Triggers](#)
- [Logging Trigger Events](#)

Introduction to Triggers

Event triggers provide a powerful way to customize ENOVIA product behavior through Program objects.

Triggers can contain up to three Programs. The triggers can work alone or together to blend functionality and control. You can write Programs that modify the behavior of many functions, and that behavior can vary depending on the circumstances. When you implement triggers you are actually creating hybrid versions of ENOVIA products. Triggers are powerful and can be used for troubleshooting problems that may arise.

Because of their complexity, you should thoroughly test event triggers in your development environment before implementing them in your production database.

Many business rules within an ENOVIA product are automated, especially rules within a business object's lifecycle. For example, when a part is promoted from Preliminary to Review, the system ensures that all the attached specifications are in the Review state and that a Find number other than the default has been specified. You can configure these automated processes by:

- Changing the argument inputs for existing rules. For example, instead of checking on the Find number, you could have the system check on a different attribute for parts, such as the classification or effectiveness date.
- Adding new rules to check. For example, you could have the system also route the part to a specific role when a part is promoted.

To view details about a trigger, you can run a trigger report against the administrative object that uses the trigger. Refer to the *Application Exchange Framework User's Guide* for instructions.



User/Password Information for a Trigger

The context.getUser() and context.getPassword() methods give the current user and password, or empty strings if there is none.

For all validate password triggers (check, override, or action), the username, password, and vault are provided as the first, second and third arguments to the trigger program, JPO or Tcl.

When the ValidatePasswordCheck or ValidatePasswordAction trigger is called, it sets USER and PASSWORD to the TARGET username/password, and args 1,2,3 to the TARGET user/password/vault. This contrasts to the MQL print context still showing the current ORIGINAL user.

If the ValidatePasswordCheck or ValidatePasswordAction trigger calls other programs (Tcl or JPO), the USER macro will be reset to represent the current (ORIGINAL) context and will remain set to that value.

When using ValidatePassword triggers, you should rely only on the input to the configured trigger program (USER/PASSWORD or args 1,2,3 to provide info on the TARGET context), and rely on 'print context' to report the current (ORIGINAL) context.

If you need to pass the target context information to a called program, or refer to it after calling other programs, capture the values in local variables in the configured trigger and to pass/reference those local variables.



User Interaction

Triggers should be used to perform behind the scenes operations on data, and should not require any interaction with the end user.

Since triggers already enlarge the event transaction, you do not want to tie up database resources any longer than necessary. For example, creating an object locks a row in the database table corresponding to the object type. To maintain database integrity, other sessions attempting to access that table must wait until the create transaction is committed. If a trigger program creates a business object, it should avoid presenting any user interface (for example, a Tk dialog box, or a wizard) since this results in locking the database table until the user dismisses the dialog.

If it is absolutely necessary for a trigger to interact with the user, this condition can be avoided by deferring the action

trigger to run outside the current transaction, by using the Execute Deferred setting in the program definition.



Lifecycle Checks and Actions with Event Triggers

Existing policies may employ check and action triggers as defined before event triggers were available. These Programs are still valid and are executed in conjunction with event triggers as explained below. ENOVIA recommends that new checks and actions are attached as triggers to take advantage of the wider functionality.

- 1) Promote event transaction is started
- 2) State1 promote Check Trigger is fired, if it exists.
 - 3) If Check blocks, then transaction aborts.
If not, State1 promote Override Trigger is fired if it exists.
- 4) If event is not replaced:
 - 4.1) Promote Access is checked --> abort if no access
 - 4.2) State transition requirements satisfaction checked --> abort if not satisfied.
 - 4.3) State transition Check program executed (OLD CHECK) --> abort if fails.
 - 4.4) Is State1 disabled? --> abort if yes.
 - 4.5) Promote object to State2
 - 4.6) State2 Notification and Routing occurs, if any.
 - 4.7) State2 Action program executed (OLD ACTION)
 - 4.8) Add promote history to object.
- 5) Promote transaction committed.
- 6) State1 promote Action Trigger fires.

State1's Action Trigger fires even though the old lifecycle action of State2 has already executed. Lifecycle actions are part of the event transaction: a failure in the lifecycle action program results in the promote transaction being rolled back. The exit code of a lifecycle action is checked, and if it returns a non-zero integer, the promotion is also rolled back.



Logging Trigger Events

When designing and testing triggers, it is highly recommended that you use a Log file so that when unexpected behavior occurs, you can trace it to the culprit program.

Refer to the *ENOVIA Live Collaboration Installation Guide* to logging details.

How Rules are Automated

The ENOVIA products automate business rules using trigger programs that run when a particular trigger event occurs.

All trigger programs must be represented by program objects in ENOVIA Live Collaboration. The programs can be Tcl or Java Program Objects (JPOs). There are three kinds of trigger programs: Check, Override, and Action. For a description of each kind of trigger program, see [Types of Triggers](#).

ENOVIA products handle triggers in a different way than in standard Studio Modeling Platform. This different approach lets you specify multiple rules for a specific trigger event. You can run multiple trigger programs of the same kind for an event (for example, run 3 check programs when a part is created). It also makes it easier to enter input for the trigger program arguments.

Make sure the MX_TRIGGER_RECURSION_DETECTION parameter is set to Signature in the initialization files (bos.ini, enovia.ini for the Live Collaboration Server).

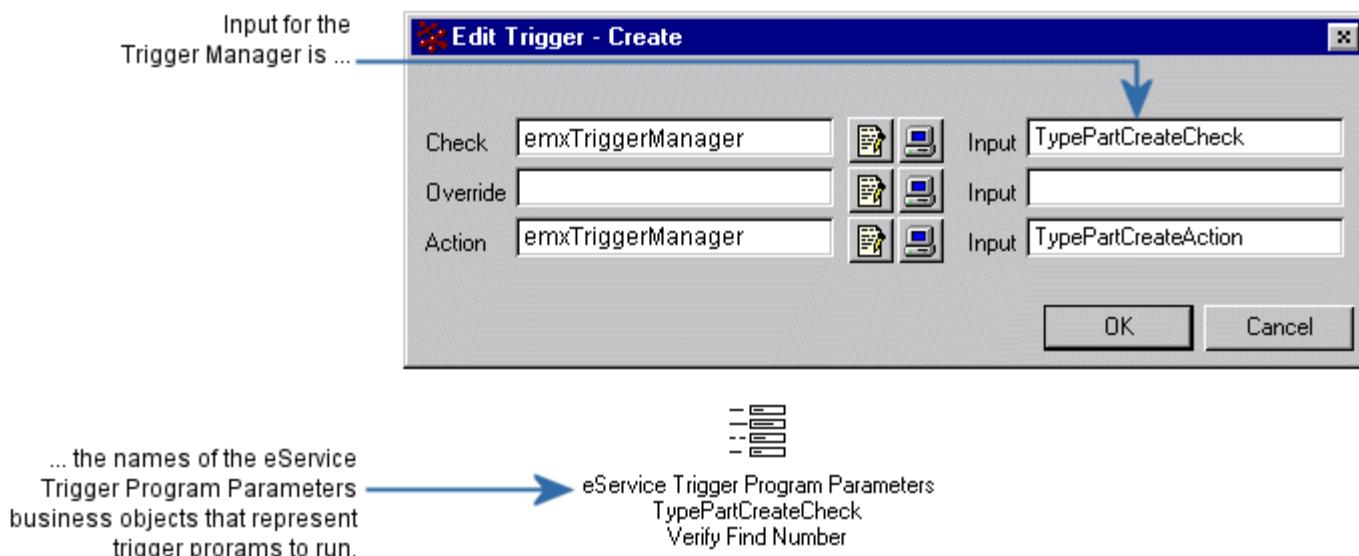
The following topics are discussed:

- [Trigger Management Components](#)
- [Trigger Program Parameters Name and Revision](#)
- [Trigger Manager Processing](#)
- [Trigger Sequence Guidelines](#)
- [Sources of Trigger Programs](#)

Trigger Management Components

The trigger management system contains the main components described in this section.

The first component, installed with Business Process Services, is the Trigger Manager (emxTriggerManager). The Trigger Manager accepts input for one argument only. This argument is the name of business objects of a type called eService Trigger Program Parameters, the second component of the trigger management system. Trigger Program Parameters objects represent specific trigger programs and are installed with the ENOVIA products.



eService Trigger Program Parameters objects have attributes that specify the:

- trigger program to run
- inputs to be passed for the trigger program's arguments
- order in which the trigger program should be executed, if other trigger programs of the same kind run for the trigger event
- if the trigger program is a Java program, the name of the method inside the JPO to run

One eService Trigger Program Parameters object must exist for each trigger program that needs to be run for a trigger event. For example, if a trigger event has two check programs and one action program, there must be three Trigger Program Parameters objects.



eService Trigger Program Parameters

TypePartCreateAction
Route part to Supervisor



eService Trigger Program Parameters

TypePartCreateCheck
Verify Find Number



eService Trigger Program Parameters

TypePartCreateCheck
Check for other parts



Trigger Program Parameters Name and Revision

To ensure the name of each Trigger Program Parameters object is unique and to help you easily identify the different objects, ENOVIA recommends using this naming convention.

[ADMIN OBJECT TYPE][ADMIN OBJECT NAME][TRIGGER EVENT][TYPE OF TRIGGER]

Using this convention, the name of a Trigger Program Parameters object indicates the trigger the object is associated with. For example, a Trigger Program Parameters object called "TypePartCreateAction" indicates that the object references an action trigger program that runs when a part is created.



eService Trigger Program Parameters

TypePartCreateAction
Route part to Supervisor

Policy triggers require one more piece of information for the Trigger Program Parameter object name:

[ADMIN OBJECT TYPE][ADMIN OBJECT NAME][STATE NAME][TRIGGER EVENT][TYPE OF TRIGGER]

For example, this Trigger Program Parameter is associated with an action trigger program that runs when an object is demoted from the Active state of a Person policy.



eService Trigger Program Parameters

PolicyPersonStateActiveDemoteAction
Notify Supplier Representative

Here are some examples of names for Trigger Program Parameters for each administrative object type:

- attributes

AttributeAddressModifyCheck

- type

TypeCompanyCheckinAction

TypeECRChangeOwnerCheck

- relationship

RelationshipDocumentsFreezeAction

RelationshipCapabilityModifyAttributeAction

- policy

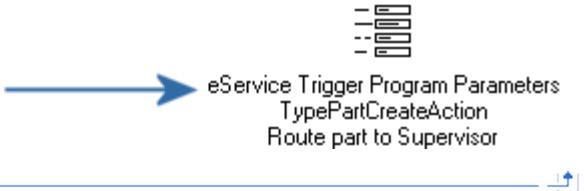
PolicyProductReleaseApproveCheck

Policy ProductReleaseOverrideAction

When a trigger event occurs, the Trigger Manager queries for all Trigger Program Parameter objects that have the same name as the input. The query includes all revisions. If more than one trigger program should execute for the same trigger event and the same kind of trigger (Check, Override, Action), the names of the Trigger Program Parameter objects must be the same and the revision must be different. The revision can be a standard number or letter, or it can be a description of the trigger program.

As an example, if two check programs should run when a part is created, there must be two Trigger Program Parameters objects with the name TypePartCreateCheck but with different revisions. Each object has attributes that reference a specific trigger program and set of argument inputs. The objects also have an attribute that specifies which program should run first.

This object references an action trigger program that runs when a part is created.



Trigger Manager Processing

When a trigger event calls the emxTriggerManager program, the program performs the steps described in this section.

1. Make a copy of all of the environment variables.
2. Set a new environment variable called ORIGINAL_INVOCATION with the value from the original INVOCATION variable.

When a Tcl trigger program is executed through the JPO Trigger Manager, this value gets set to program and you do not want to lose the original INVOCATION value. This means that triggers that rely on the INVOCATION environment variable for program execution should look for the ORIGINAL_INVOCATION environment variable.

3. Query for business objects that meet this criteria:

Name--Input specified for the emxTriggerManager program. This is the name of the eService Trigger Program Parameters object that represents the trigger program to be run.

Type--eService Trigger Program Parameters

Revision--* (all)

Vault--eService Administration if there is such a vault or * (all) if eService Administration does not exist.

4. Discard all returned objects that are in the Inactive state.
5. Sort the objects in ascending order according to the eService Sequence Number attribute.
6. Loop through each of the objects found and determine if they are Tcl or Java programs.
7. Execute the programs in the specified sequence and include the inputs. Trigger Manager waits for each program to complete successfully before calling the next program. If trigger is a Java program, get the name of the method inside the JPO and invoke it. The method is specified in the eService Method Name attribute for the eService Trigger Program Parameters object. If no method is specified, default to the main (mxMain).

Or:

If the trigger is a Tcl program, execute the program using the MQL command class.

8. The program returns zero for success and a nonzero for a failure.

If, after invoking a JPO trigger, the Trigger Manager gets a return value of 1 and then checks the INVOCATION type. If the INVOCATION is an action (Action Trigger), it throws an exception. For any other type of INVOCATION, the Trigger Manager JPO returns 1 to the invoking environment to indicate failure. Otherwise, a zero is returned to indicate success.

For each trigger that is completed without error, the Trigger Manager resets all of the environment variables back to their original values, until all trigger(s) are executed successfully.

If any program returns failure, the Trigger Manager stops execution and returns failure. In the case of a check trigger, failure prevents the event from occurring. In the case of an action trigger, the programs that are called do not return success or failure because actions do not wait for a return code. If a program that is called as an action fails as a result of an error in an MQL command, the event that invoked them will, as expected, be aborted. However, if you want to trap an error in an action program by means other than MQL errors, you can use something like the following:

```
If {sString == ""} {
    mql abort transaction
    return
}
```

Trigger Sequence Guidelines

The guidelines for running one trigger before or after another trigger are listed in this section.

1. Run Triggers that are expected to fail more often earlier than triggers that are not expected to fail as often. This reduces the amount of time a user waits for triggers that will usually pass because they will only be run after all of the other more likely-to-fail triggers have passed.
2. Run fast Triggers before slow triggers. It is better to get through fast triggers early so if the event must be attempted more than once, the user will not spend as much time waiting for a slow trigger to run more times than necessary.
3. Triggers that perform checks which other triggers (within the same event) depend on must be run first so that those preconditions are satisfied for the later dependent triggers. This reduces redundant checking of subsequent triggers because all preconditions have been met.

Items #1 and #2 above are subjective and require decisions based upon domain knowledge of the process and data to make the proper trade-offs.

Item #3 indicates that triggers installed with the framework and applications should not have their order changed relative to each other unless you know there are no interdependency checks. To determine if there are interdependencies, see the Administrator's Guide for the application.

Sources of Trigger Programs

Trigger programs can come from the main sources listed in this section.

- The trigger programs used by many applications, called utility trigger programs, are installed with Business Process Services and are described in "Utility Trigger Programs" in the *Schema Reference Guide*.
- Programs specific to an ENOVIA product are installed with the application. For information about trigger programs that are specific to an application, refer to the administrator's guide that accompanies the ENOVIA product.
- The Subscription Services Application Library contains many trigger programs.
- You can create your own custom trigger programs.

Do not make changes to the programming for trigger programs that are installed with Business Process Services or ENOVIA products. Handle any adjustments using the argument inputs passed by the eService Trigger Program Parameters object and described in [Modifying Inputs Trigger Programs](#).

Do not change the file names of the trigger programs. The file names for all ENOVIA products programs are prefixed with "eService[name of installed directory]". Do not change this prefix and do not create custom programs using the eService prefix. If you need to modify the name of an ENOVIA product program, clone the program and change the name of the cloned program.

Trigger Scenarios

With the power of Event Triggers, it is possible to programmatically guarantee certain events cause (or are replaced by) other actions. Some scenarios for triggers are listed below.

- An Attribute range could be updated upon creation of an object. For example, when a Vendor object is created, its name is added to the "Provider" Attribute range.
- When an Attribute is modified, a connection could occur. For example, when the Provider Attribute field of a Purchase Order object is entered, the P.O. could be connected to the Vendor object.
- Relationships could be given a lifecycle. For example, three relationships could exist called "Proposed", "Planned" and "Complete." A Project object could be connected to various Tasks with the Proposed relationship. When the Task has been worked out, the two objects are connected with the Planned relationship, which triggers a disconnect of the Proposed relationship. Likewise, when the Complete connection is created, the Planned connection is removed automatically.
- The changename event could trigger the name of a related object to be changed. For example, if a Product name is changed, the Manual object which will document it could be automatically changed, and notification sent.
- The checkout event could trigger a program that would check to see if the object is locked and prompt the user to lock if they want to check the file back into the object.
- The checkin event could trigger the object to be unlocked.
- When objects are deleted, if files exist, they can programmatically be copied to a specified location or externally be emailed to the System Administrator.
- When one state is scheduled, each successive state could be scheduled automatically by a specified offset value.
- Objects could be added or removed from sets when an event occurs. For example, the set called "Pending Projects" could be modified as objects reach the "Complete" state.

Designing Triggers

This section provides the background needed to write triggers.

In this section:

- [Supported Trigger Events](#)
- [Types of Triggers](#)
- [Multi-trigger Events](#)
- [Transaction Boundaries](#)
- [Notification API and Triggers](#)
- [Modifying Inputs Trigger Programs](#)

Supported Trigger Events

Most of the events associated with Business Objects, Relationship Objects, Workflows, Policy lifecycle, and Transaction events support Triggers.

Triggers are added to Type, Relationship, Attribute, Process, and State definitions in much the same way as Attributes and Methods are added to the Type definition. When used in Types, they are likewise inherited.

The following topics are discussed:

- [About Inheritance](#)
- [Business Object Events](#)
- [Attribute Events](#)
- [General Modify Events](#)
- [Relationship Events](#)
- [State Events](#)
- [Transaction Events](#)
- [Workflow Events](#)

About Inheritance

In a situation where a Child type both inherits a trigger and defines a trigger on the same event (the native trigger), the native trigger is executed, and is inherited by any further Child type.

The ENOVIA Integrations Exchange Framework (IEF) and collaborative products run into this situation, as described in the scenario below

1. Business Process Services is installed and defines the CAD Drawing type with several triggers, including 1 on the Revise event.
2. The IEF is installed and defines the MCAD Drawing type, derived from the CAD Drawing type and also defines its own revise event trigger.
3. An integration such as ProE is installed, and defines the ProE Drawing type, derived from the MCAD Drawing type.
4. The revise event trigger that the ProE Drawing type inherits is the one defined in the IEF (for the MCAD Drawing type) not the one from Business Process Services assigned to the CAD Drawing type.



Business Object Events

The following Business Object events support Triggers. These Triggers can be added to Type definitions.

addinterface	connect	modifyattribute
changename	copy	modifydescription
changeowner	create	removefile
changepolicy	delete	removeinterface
changetype	disconnect	revision
changevault	grant	revoke
checkin	lock	unlock
checkout		

Refer to the "Working With Types" chapter in the MQL Guide for more information and syntax for adding triggers to Type definitions.

See [Supported Trigger Events](#) for details about using these trigger events.



Attribute Events

Attributes can have triggers on the modify event.

You can define triggers on business object or relationship modifyattribute events, which are fired whenever **any** attribute is modified (once per attribute modified), or on an attribute's modify event, which is fired when that attribute in particular is modified.



General Modify Events

You can configure a trigger for a "general" modify event of a businessobject or relationship.

When defined on a type, whenever a modification is made to any attribute or the description of a business object of that type, the program associated with the trigger is invoked at the end of the outside transaction. When defined on a relationship, modifications to any attribute on a connection of that relationship type cause the trigger to fire. This is in addition to any modifyattribute or modifydescription triggers when those events occur.

The "general" modify event supports only Action triggers; it cannot be configured with a Check or Override trigger.

The trigger program is invoked once per object/relationship per transaction. If multiple changes are made within a transaction, the trigger program only runs once. If a transaction operates on several objects, the trigger program will be run once for each object.

The programs associated with modify triggers must be JPOs. In addition, these programs will always run as deferred. If configured to run immediately, that setting is ignored and program execution is still deferred.

Because the program is deferred, if any errors occur during the program execution, all prior changes within the transaction boundary are not rolled back. In addition, the trigger program should not trigger another modify event on the same business object or connection; if it does, an error will be thrown.

The trigger for the modify event supports these generic macros:

- \$USER
- \$trigger_user

Macros specific to the business object or relationship are not available to the trigger program. In addition to the above macros, these arguments are passed to the JPO:

- \$ID--the business object ID or the connection ID
- \$IDTYPE--Type ID or Relationship ID
- \$TRANSHISTORY--history items for operations in the current transaction

\$TRANSHISTORY is a carriage return delimited string listing all changes to the business object within the transaction. Programs should process the output of this macro in the same manner as output from "print bus T N R select history".

History information is passed to the trigger program even if history is turned off.

History information for workflow events is not part of \$TRANSHISTORY.

There are several options when designing triggers for business object or connection modifications. The tables below point out the differences.

Business Object Event	Configuration	Usage Notes
modify (business object event)	Configured on Types; only Action programs can be configured.	<ul style="list-style-type: none"> • Triggers when any attribute or the description of a business object is modified. • Fires only once per transaction, no matter how many attributes (or description) are modified. • Different/smaller set of Macros are available compared to modifyattribute. • Always deferred. • Programs must be of type Java. • Designed for use with notifications in ENOVIA products.
modifyattribute	Configured on Types; supports Check, Override and Action programs.	<ul style="list-style-type: none"> • Triggers when any attribute of a business object is modified. • Fires once for every attribute that is modified.
modifydescription	Configured on Types; supports Check, Override and Action programs.	Triggers when the description of a business object is modified.
modify (attribute event)	Configured on Attributes; supports Check, Override and Action programs.	Triggers when that specific attribute is modified, whether on a business object or connection.

Relationship Event	Configuration	Usage Notes
modify (Relationship event)	Configured on relationships; only action programs can be configured.	<ul style="list-style-type: none"> • Triggers when any attribute of a connection is modified. • Fires only once per transaction, no matter how many attributes are modified. • Smaller set of macros are available. • Always deferred. • Programs must be of type Java.
modifyattribute	Configured on relationships; supports	

	check, override and action programs.	<ul style="list-style-type: none"> Triggers when any attribute of a connection is modified. Fires once for every attribute that is modified.
modify (Attribute event)	Configured on attributes, supports check, override and action programs.	Triggers when that specific attribute is modified, whether on a business object or connection.
modifyfrom	Configured on relationships	Triggers when the object connected to the from end of the relationship is modified
modifyto	Configured on relationships	Triggers when the object connected to the to end of the relationship is modified

Relationships additionally support the modifyfrom and modifyto events to address the specific scenarios of how objects on either end of a relationship can be replaced by another object, and as such they do not affect attribute or description modification.

The modifyto and modifyfrom events have these macros available to them:

Macros from the Parent Event (modifyconnection)	Macros from the Original Object	Macros from the New Object
RELTYPE	FROMOBJECT	OBJECT
DESCRIPTION	FROMTYPE	TYPE
FROMMEANING	FROMNAME	NAME
FROMTYPES	FROMREVISION	REVISION
FROMALLFLAG	FROMOBJECTID	OBJECTID
FROMREVACTON	TOOBJECT	DESCRIPTION
FROMCLONEACTION	TOTYPE	OWNER
FROMCARD	TONAME	LOCKER
TOMEANING	TOREVISION	LOCKFLAG
TOTYPES	TOOBJECTID	CURRENTSTATE
TOALLFLAG		NEWRELID
TORREVACTON		
TOCLONEACTION		
TOCARD		
USER		
RELID		
ISFROZEN		
VAULT		
TRIGGER_VAULT		
LATTICE		
CHECKACCESSFLAG		
CONNECTION		



Relationship Events

The following relationship events support Triggers, and can be added to Relationship definitions.

freeze	create (connect)	modifyattribute
thaw	delete (disconnect)	

Refer to the "Working with Metadata" chapter of the *MQL Guide* for more information and syntax for adding triggers to Relationship definitions.

See [Supported Trigger Events](#) for specific information about using these trigger events.



State Events

The following lifecycle events support Triggers. These Triggers can be added to Policy definitions.

approve	demote	disable
enable	ignore	override
promote	reject	schedule
unsign	?	?

Refer to "Working with Metadata" chapter of the *MQL Guide* for more information and syntax for adding triggers to Policy definitions.



Transaction Events

You can configure a transaction trigger that will be invoked one time for all events on a type or relationship at the end of a transaction.

You can configure a transaction trigger that will be invoked one time for all events on a type or relationship at the end of a transaction. When defined on a type or relationship, whenever a business object or connection of that type or relationship is modified, at the end of the outside transaction, the program associated with the transaction trigger is invoked.

If both a transaction trigger and a general modify trigger are defined for a type, both triggers will be invoked at the end of the outside transaction.

Transaction triggers can only be Action triggers; and cannot be configured as a Check or Override trigger.

The programs associated with transaction triggers must be JPOs. In addition, these programs will always run as deferred. If configured to run immediately, that setting is ignored and program execution is still deferred.

Because the program is deferred, if any errors occur during the program execution, all prior changes within the transaction boundary are not rolled back. In addition, the trigger program should not trigger another transaction event on the same business object or connection; if it does, an error will be thrown.

Transaction trigger JPOs are not invoked for objects that are deleted inside a transaction.

Macros specific to the type or relationship are not available to the trigger program. Only one macro can be passed as an input parameter to a program for a transaction trigger:

- \$TRANSHISTORY--history items for operations in the current transaction

\$TRANSHISTORY is a carriage return delimited string listing all changes to the business object within the transaction. Programs should process the output of this macro in the same manner as output from "print bus T N R select history". The string size limit for this macro is 2^31-1 bytes. If this limit is exceeded, the program call is split and multiple calls may be made depending on the amount of history information.

Each time a transaction trigger is invoked, information on all events that have occurred is passed to the program. For example, if business object 1 is promoted and the owner of business object 2 is changed, at the end of the transaction, the transaction trigger is invoked on time and \$TRANSHISTORY is populated.

History information is passed to the trigger program even if history is turned off.

History information for workflow events is not part of \$TRANSHISTORY.

The following business object events support transaction triggers:

addinterface	connect	modifyattribute
changename	copy	modifydescription
changeowner	create	removefile
changepolicy	delete	removeinterface
changetype	disconnect	revision
changevault	grant	revoke
checkin	lock	unlock
checkout		

The following connection events support transaction triggers:

create	addinterface	modifyto
delete	modifyattribute	removeinterface
freeze	modifyfrom	thaw



Workflow Events

The following Workflow and Activity events support triggers. These Triggers are defined in the Process and Activity objects.

Process events

FinishProcess	ReassignProcess	ResumeProcess
StartProcess	StopProcess	SuspendProcess

Auto Activity Events

ActivateAutoActivity	CompleteAutoActivity	OverrideAutoActivity
ReassignAutoActivity	ResumeAutoActivity	SuspendAutoActivity

Interactive Activity Events

ActivateActivity	CompleteActivity	OverrideActivity
ReassignActivity	ResumeActivity	SuspendActivity

Refer to the "Working with Workflows" chapter of the *MQL Guide* for more information on adding triggers to processes and activities.

Types of Triggers

Each supported data event has three types of triggers.

- Check trigger that fires before the event occurs
- Override trigger that can replace the event transaction
- Action trigger that fires after the event occurs

For each event, Programs can be specified for none, any, or all of the Trigger types.

The following topics are discussed:

- [Check Trigger](#)
- [Override Trigger](#)
- [Action Trigger](#)

Check Trigger

The check trigger executes first, before any of the normal system code which carries out the event has a chance to execute.

The program specified here can be used to:

- Perform pre-event processing in anticipation of the event
- Block the event from actually occurring, aborting the event transaction, which means that no other trigger programs are get executed.

Refer to [Trigger Programs](#) for more information.



Override Trigger

The override trigger fires after the check trigger, but before any of the normal system code that carries out the event has a chance to execute.

It can also be used to:

- Perform pre-event processing after any check program has decided not to block the event.
- Replace the normal event processing with an alternate event.
- Perform a soft block on the event., meaning that the event is blocked without raising an exception, the action program is executed and any pre-processing done in the check program is not rolled back. Refer to [Trigger Programs](#) for more information.

Refer to [Trigger Programs](#)for more information.



Action Trigger

The action trigger fires after the event transaction is committed, whether the normal event took place or the override program replaced the event.

The action program typically extends the functionality associated with an event (or the override program that replaced the event). A program attached to the action trigger can perform any post-event processing that must be done after the event has updated the database. For example, an action program may generate a report in an external application that lists the changes that have just occurred. Since you would not want this action to fire unless those changes actually took place, the action trigger is always fired after the event transaction is committed.

Consider, however, that the same event is part of a larger transaction. If the action trigger fired immediately after the event, generating a report outside of the ENOVIA Live Collaboration, what would happen if the transaction failed? The ENOVIA Live Collaboration can only roll back internal state changes, so the report would be generated erroneously. In this case, the action program execution should be defined as Deferred so that it is deferred until all transactions are committed.

Refer to [Trigger Programs](#) for more information.

Multi-trigger Events

When designing Triggers, keep in mind that several Triggers may be fired by what appears to be a single event. For example the "Open for Edit" Event appears to the user to be one operation, but under the covers, the system performs four Events (checkout, lock, checkin, and unlock), each of which may have three Triggers fire (check, override, action). Carefully evaluate exactly where (which event and trigger) their logic should be introduced. Remember that if programs exist in many of these Triggers, performance will be affected.

The following topics are discussed:

- [Two Phases of Create Event](#)
- [Relationship Triggers](#)

Two Phases of Create Event

You can create a business object in either MQL (with the `add businessobject` statement) or within Matrix Navigator (using the Object/New menu selection).

You can create a business object in either MQL (with the `add businessobject` statement) or within Matrix Navigator (using the Object/New menu selection). In both cases, it is a 2-phased process: the creation phase, then the modification phase. Likewise, creating clones or revisions (`copy businessobject` or `revise businessobject`) as well as connections (`connect businessobject`) are performed in 2 phases.

When designing triggers, it is important to understand what occurs when, since a non-deferred program may require data that has not yet been entered into the database. The following example describe the `businessobject` creation event, and also applies to connections.

At creation, the object is given a type, name, revision, policy, and vault. Any non-deferred action program assigned to the initial state of the object executes, followed by the non-deferred create action program (if any). Since the object has not yet been given anything more than an identity, only very basic macro values (see [Macros](#)) have any meaning at this point, and so these action programs must rely only on the information available.

In the modification phase, the description, attribute values, owner, and schedule information (if any, from MQL) are defined. During this phase many events may occur. There will be a `modifyattribute` business object event for each attribute (as well as a `modify` event on the attribute) having its value set. There may also be a `changeowner` event, `modifydescription` event, and any number of schedule events. Thus, any check, override, and non-deferred action programs associated with these types of triggers are executed during this modification phase. Because of this, action programs which at first seem appropriately placed on the create event, may actually work well on another event that occurs during the modification phase.

Another aspect of the create event has to do with transaction boundaries. Transaction boundaries for creating business objects differ depending on which Studio Modeling Platform application is used (Matrix Navigator or MQL). The create and modification events are one transaction in MQL, and split into two in ENOVIA Live Collaboration. In either case, if the creation phase fails the business object is not created. However, if the modification phase fails, in ENOVIA Live Collaboration the object will still exist; in MQL it will not exist. If the execution of any action programs is deferred until after the outer transaction commits, they will behave differently when using MQL and the ENOVIA products:

- In the ENOVIA products, any deferred state action program and/or create action trigger executed before the modification phase.
- In MQL, they execute after the modification phase.

In summary, when writing programs for use in the create and modify business object events, keep in mind the following:

- The object creation happens in two phases: the creation of the object, then the modification of the object's attributes. During the creation phase, the following occurs:
 1. The create check trigger program executes.
 2. The create override trigger program executes.
 3. The create event occurs.
 4. Any action defined on the first state in the lifecycle executes.
 5. The non-deferred create action trigger program executes.
- Programs written as create event action triggers must use only those macros that pass basic information since the object knows only its type, name, revision, policy, and vault. Place Programs that require more information as triggers on another event that occurs during the modification phase.
- The create and modification events are one transaction in MQL, and split into two in the Studio Modeling Platform, and therefore deferred action programs execute after different phases depending on which application is used.



Relationship Triggers

When forming or breaking a relationship between two business objects, Trigger programs are fired in the following order:

1. The connect or disconnect Check of the FROM Business Object

2. The connect or disconnect Check of the TO Business Object
3. The create or delete Check of the Relationship
4. The create or delete Override of the Relationship
5. The connect or disconnect Action of the FROM Business Object
6. The connect or disconnect Action of the TO Business Object
7. The create or delete Action of the Relationship

Notice that the Override Triggers on the FROM and TO Objects are NOT fired.

Important: Override Programs specified in Business Object Connect/Disconnect Events are NEVER fired. The Override Program required for establishing or breaking relationships should be specified in the Relationship create/delete Trigger.

Consider the consequences of the connection operation being replaced at one end (by an Override program) but not at the other. The resulting Relationship is invalid, since it would be missing an end. For this reason, and to avoid unnecessary complexity, only one Override program is available for create/delete connection events. Since Relationship Trigger programs have access to macros that can provide information about both connection ends, one Override is enough.

Transaction Boundaries

When designing triggers, you need to understand how events and triggers interact with transaction boundaries.

The following topics are discussed:

- [Events](#)
- [Triggers](#)

Events

When designing Triggers, you must understand the ENOVIA Live Collaboration transaction model.

The discussion below is supplemental to the description in the of the *MQL Guide* chapter "The MQL Language".

Over time an object may transition from one state to another, as its properties are changed by events. The use of the term *state* here is more general than the States in an Policy. For example, changing the description of a business object is an event that changes the state of that business object, but the lifecycle State probably remains the same.

In the Studio Modeling Platform, each event is implicitly wrapped in a transaction boundary and the achievement of a new state is not complete until the transaction is committed (thus ensuring database consistency). Common Studio Modeling Platform functions are often actually a series of actions that rely on the proper execution of all of the previous actions. For example, checking access and logging history are actions that are implicitly wrapped in the transaction boundary of most Studio Modeling Platform operations. The design guarantees that objects are not updated without the history being logged.

ENOVIA Live Collaboration also allows transaction boundaries to be explicitly defined so that you can roll back all related actions if one of them fails. When transaction boundaries are implicit, the state of an object is considered changed as soon as an event transaction takes place. However, for an explicitly-defined transaction to be effective, each of its commands must be successful.



Triggers

When you add event triggers to the transaction model, the internal logic looks like this:

- 1) Event transaction is started
- 2) Normal processing of access checking occurs.*
- 3) If it exists, the Check Trigger is fired.
- 4) If Check blocks, then transaction aborts.
 If not, the Override Trigger is fired if it exists.**
- 5) The Event transaction is then committed regardless of an override or normal activity.
- 6) Finally, if there is an Action Trigger, it is fired.

*Access is checked but the event is not stopped if no access. The RPE CHECKACCESSFLAG and ACCESSFLAG are set so the check and override events can determine current state and correct action.

**The lack of access will stop the event if the override does not override the event with some other operation.

If an event is part of a larger transaction, action programs may be defined with the Execute Deferred flag so that execution is deferred until after all transactions are committed. The transaction model looks like this:

- 1) Larger transaction is started with any number of commands before the Event with the trigger.
- 2) Event transaction is started.
- 3) Normal processing of access checking occurs.
- 4) If it exists, the Check Trigger is fired.
- 5) If Check blocks, then transaction aborts.
 If not, the Override Trigger is fired if it exists.
- 6) The Event transaction is committed regardless of an override or normal activity.
- 7) If the Event Trigger has a non-deferred Action Program, it is executed.
- 8) The larger transaction is committed if all activities and triggers succeed.
- 9) If the larger transaction successfully commits, and there is a deferred Action Program, it is now triggered. If the larger transaction aborts, deferred programs are not executed.

Deferred action programs will queue up if more than one is deferred within a transaction. This ensures that the order of execution remains the same as the Events which triggered them.

Notification API and Triggers

The emxNotificationUtil JPO and emxTransactionNotificationUtil JPO expose several public methods that can be called through the Trigger Manager or directly from a bean or JPO. The emxTransactionNotificationUtil JPO is distinct in that it is invoked only after a transaction completes.

For instructions on using the Trigger Manager to define email notifications, see the *Live Collaboration Administrator's Guide*.

You use the emxNotificationUtil JPO in conjunction with a custom JPO and the Notification object as an alternative to emxTriggerManager as the program object for an action trigger. You would use emxTransactionNotificationUtil JPO in conjunction with a transaction trigger.

The emxNotificationUtil JPO contains these methods:

Method Name	Static	Return Type	Argument Passed	Called From
objectNotification	Yes	int	(Context context, String[] args)	Trigger
objectNotification	Yes	void	(Context context, String objectid, String notificationName, Map payload)	JPO
objectNotificationFromMap	Yes	int	(Context context, String[] args)	Bean
relationshipNotification	Yes	int	(Context context, String[] args)	Trigger
relationshipNotification	Yes	void	(Context context, String relid, String notificationName, Map payload)	JPO
relationshipNotificationFromMap	Yes	int	(Context context, String[] args)	Bean

To call a method from a trigger, the args array contains two entries:

- objectid *or* relid
- name of the notification object

See the *Live Collaboration Administrator's Guide* for instructions on configuring notification objects.

emxTransactionNotificationUtil requires the setting, "History Bit," to be configured on the commands that are defined for those events to which users can subscribe. See "Working with Commands and Menus" in the Business Modeler Guide.

To call a method from a JPO, pass the id and notification name as separate arguments. You can also include an optional payload map argument to pass information to the JPO methods required to create and send the notification.

To call a method from a bean, pass a single map entry as the args array. The map contains:

- objectid *or* relid
- notification name
- optional map argument to pass information to the JPO methods required to create and send the notification

Modifying Inputs Trigger Programs

You can change the argument inputs that are passed to a trigger program and the sequence in which the program is executed by modifying the attributes for the program's eService Trigger Program Parameter object.

These procedures describe how to add triggers using Matrix Navigator and Business Modeler, but you can use MQL instead.

1. In Matrix Navigator, find the eService Trigger Program Parameters object that represents the trigger program.
2. Access the object's attributes.
3. Modify the eService Program Argument attribute that contains the input you want to change.

Make sure you use the symbolic name for any administrative objects that you specify as an input. For example, if you need to specify a relationship for an argument input, use the symbolic name for the relationship and not the standard name (relationship_GBO instead of GBO). For a list of accepted argument inputs for a trigger program and the correct order for the arguments, refer to "Utility Trigger Programs" in Appendix B of the *Schema Reference Guide* and to the guide that accompanies your ENOVIA product.

4. Change the number for the eService Sequence Number attribute so it represents the order in which the program should run. For example, if three action programs execute when the trigger event occurs and this program should run last, enter 3. If it does not matter which order the programs run in, you must still enter a numeric value other than 0.

Trigger Programs

You need to understand the types of trigger programs.

Note the following about Program objects attached to Event Triggers:

- Any changes made to the Program object are immediately picked up when fired by a trigger. This implies that a check or override program can modify the code of an action program that follows.
- Check and Override programs must return an exit code in order for the event processing to know how to proceed. Code can be included in the Program that performs other functions prior to setting the exit code's integer value.
- Programs that use trigger-specific macros cannot be used as methods, since the macro values rely on the triggering event.
- When a Program object attached to a trigger is renamed, the newly-named Program object remains attached to the trigger.
- Deleting a Program object that is attached to a trigger automatically detaches the Program object from the trigger.

The following topics are discussed:

- [Trigger Macros](#)
- [Check Programs](#)
- [Override Programs](#)
- [Action Programs](#)

Trigger Macros

Macros are a simple name/value string substitution mechanism, where the macro value is substituted for the macro name when used in a Program as follows:

`$ {MACRONAME}`

This single one-pass string substitution process occurs just prior to program execution. Refer to [Macros](#) for more details



Check Programs

Do not defer trigger check program execution. Make sure the exit code is an integer value that is always tested by the trigger logic to determine whether or not to block the event. Any system errors encountered while executing the check program will abort the event transaction. When the check program returns a non-zero exit code, the event is not performed. This is one way to block events from occurring.

Unlike replacing events (see below), blocking events with a check program causes the event transaction to abort, and raises an exception (and therefore displays an error). If a check program blocks, then neither the override program nor the action program get executed.

Events can also be blocked using an override program. To block an event without having an error message presented, use an override trigger. If the exit code from the override program is a non-zero integer, the event is completely replaced, but the transaction is committed, and no error message is displayed.

In this scenario the normal event processing is skipped, and therefore the override program *becomes* the event. If the override program causes the event to be replaced without providing code for an alternative event, the original event is, in a sense, blocked. This is called a soft block. It is different from blocking with the check program in that the event transaction is committed anyway. Since the transaction is not aborted, any processing that may have occurred in the check program is not rolled back and the action trigger will fire. If there is an action trigger program, you may need to use the RPE to communicate to the action program that the event was blocked, and not to do its normal activity.



Override Programs

Do not defer override program execution. The exit code should be an integer value, and is always tested by the trigger logic to determine whether or not to replace the event.

If the override program returns a zero exit code, the event is not replaced, and the flow of control proceeds in its normal fashion. That is, the normal system code is executed, the history of the event is recorded, the transaction is committed, and then the action trigger is fired, if it exists. You can use this technique to add pre-event processing to an ENOVIA Live Collaboration event.

Replacing Events

When the override program returns a non-zero exit code, the normal system event code and history mechanism is skipped, and then the event transaction is committed and the action program is executed.

When the override program returns a non-zero exit code, the normal system event code and history mechanism is skipped, and then the event transaction is committed and the action program is executed. This method lets you replace existing event handling with new, custom event handling. A non-zero exit code in an override program can also be used to block the event.

Refer to [Override Programs](#).

When replacing normal event processing, the associated history entry is not recorded. This brings up the issue of which events should occur in the override program. Certainly if the overriding behavior causes other *normal* ENOVIA Live Collaboration events to occur, those events are logged by their associated history mechanism. If this is not the case, the overriding behavior is unrecorded by the history mechanism. You can use other means of recording the event such as the check or action code. The following MQL command can be used to add custom history entries to a business object:

```
modify bus TYPE NAME REV add history VALUE [comment VALUE];
```

For details, see "Extending an Application" in the *MQL Guide*.

Changing the behavior of the ENOVIA products without any explicit indication, as is the case when using the replace event feature, can lead to problems that are extremely difficult to troubleshoot. Therefore, when triggers that are designed to replace expected ENOVIA behavior are implemented, it is recommended that you create a trigger log by setting the environment variable `MX_TRIGGER_LOG` to `TRUE`.

Any system errors encountered while executing the override program will abort the event transaction.

Override on Disconnect

A delete relationship (disconnect) override program poses a special case, since this event occurs not only when 2 objects are simply disconnected, but also when an object on 1 end of a relationship is deleted. Before ENOVIA Live Collaboration deletes an object from the database, any connections it has are first removed, which results in an invocation of the delete relationship triggers.

If the override program returns an exit code of 1 (implying that the trigger took care of the deletion), ENOVIA Live Collaboration will double check to insure that the relationship was really removed. If it was not, it aborts the current transaction (that attempted to delete the business object) and issues the following error messages:

```
Error: #1900068: del business object failed
System Error: #1500573: Override trigger for deleting
connections failed to delete relationship RELID when business
object at one end was being deleted.
```

This error indicates that the override trigger program is incorrectly coded, and should be reexamined to provide the required functionality (perhaps rewriting as an action program.)



Action Programs

Without transactions and potential rollbacks to worry about, it is best to execute the action program immediately after the event takes place. This is the case with non-deferred action programs.

Refer to [Types of Triggers](#) for more information.

Trigger action program execution is deferred if the Execute Deferred setting is on. Deferred action programs are queued for execution which takes place when the outer-most transaction is committed. See the "Extending an Application" chapter of the *MQL Guide* for additional information.

Trigger action program exit codes are never tested by the trigger logic. (This is not the case, however for lifecycle actions. In a state promotion action, if the program returns 1 or aborts, the promote transaction is aborted.)

Any system errors encountered while executing a non-deferred action program will only abort the event transaction if it is nested within an outer transaction.

Adding a Trigger for a Trigger Event

This section describes how to run a trigger program when a particular trigger event occurs. The instructions assume the trigger program has already been installed (or created if you have a custom program).

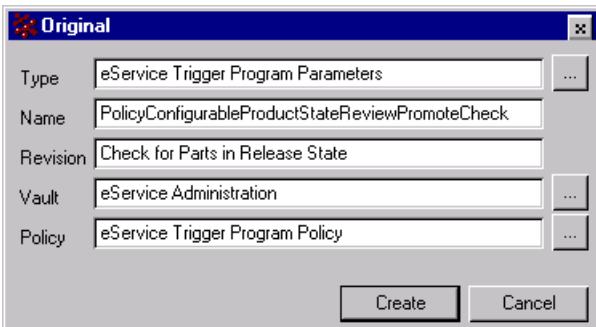
Make sure you follow the instructions carefully to avoid making these common mistakes:

- Creating the eService Trigger Program Parameters object in a vault other than eService Administration.
- Failing to promote the eService Trigger Program Parameters object to Active after creating it.
- Not matching the name of the eService Trigger Program Parameters object to the name entered in the Input box for the trigger in Business Modeler.

These procedures describe how to add triggers using Matrix Navigator and Business Modeler, but you can use MQL instead.

1. In Matrix Navigator, create an eService Trigger Program Parameters object that represents the trigger program.

- a. Select **Object>New>Original**.
- b. Complete the Original dialog box.



Type--eService Trigger Program Parameters.

Name--The name should indicate the trigger the object is associated with by including the administrative type (policy, type, attribute), the administrative object name (Part, Originator, etc.), the trigger event (Create, Promote, etc.), and the kind of trigger program (Check, Override, Action). For policies, include the state name also. For more information, see [How Rules are Automated](#).

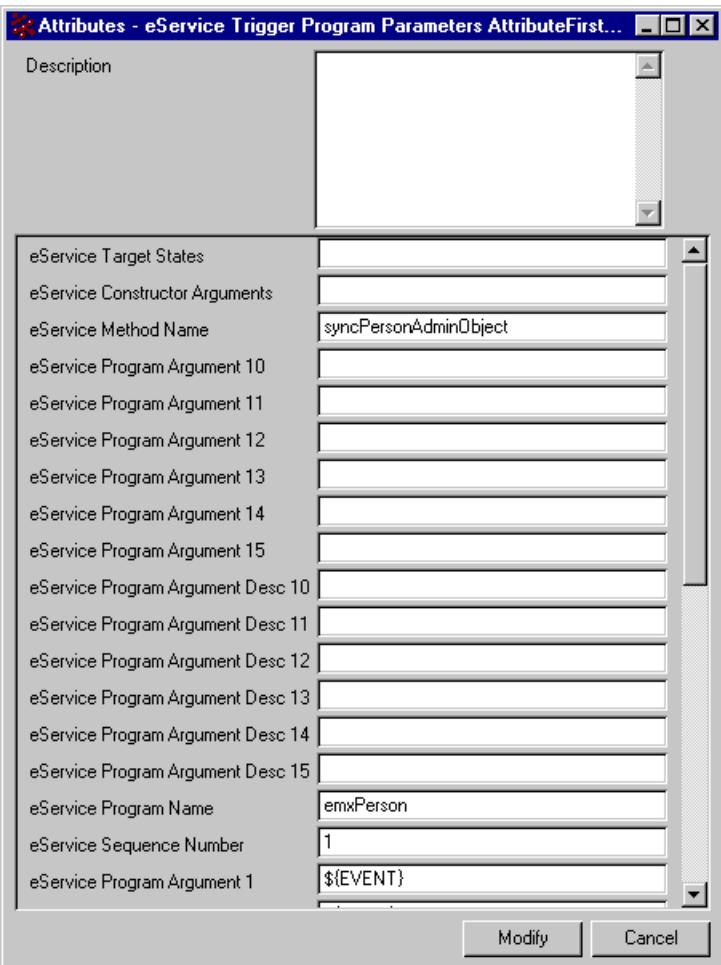
Revision--Any string, such as 1 or A or a. You may want to include a description of the trigger program associated with the object.

Vault--The vault must be eService Administration.

Policy--eService Trigger Program Policy.

- c. Click **Create**.

2. Enter attributes for the new object.



eService Program Name--Enter the name of the program object that should be executed when the trigger event occurs.

eService Sequence Number--Enter a number that represents the order in which the program should run. For example, if three check programs execute when the trigger event occurs and this program should run last, enter 3. If it does not matter which order the programs run in, you must still enter a numeric value other than 0. For more information, see [How Rules are Automated](#).

eService Program Argument *n*--Enter a value for each argument the object should pass to the trigger program, making sure the argument inputs are in the correct order. Also make sure you use the symbolic name for any administrative objects that you specify as an input. For example, if you need to specify a relationship for an argument input, use the symbolic name for the relationship and not the standard name (relationship_GBOM instead of GBOM). For a list of accepted argument inputs for a trigger program and the correct order for the arguments, refer to "Utility Trigger Programs" in Appendix B of the *Schema Reference Guide* and to the guide that accompanies your ENOVIA product.

Remember to register the symbolic name for any custom administrative objects (objects that are not supplied with the framework). If you use an unregistered symbolic name as an input for a program, the program will not work. See "Registering Your Own Administrative Objects" and "Configuring State Names" in the *Application Exchange Framework User's Guide*.

eService Program Argument *n* Description--Enter a description for what each argument does. This description can also include other details, such as the default value for the argument.

eService Method Name--The name of the method to invoke in the JPO. Specifying a method is important if you need to use the same JPO with multiple methods in many different triggers. If no value is entered, the Trigger Manager uses the main (mxMain) method of the JPO.

eService Target States--Only used for promote and demote triggers. Stores a comma separated list of the symbolic names of states. The Trigger Manager fires the trigger program only when promoting or demoting to one of the listed states. If the target state is not in the list, the trigger does not fire. If the attribute is not populated, the Trigger Manager fires the program regardless of the target state.

This attribute is useful for policies that allow branching to multiple states but the trigger program should not run for all states. For example, suppose a policy allows an object to be promoted from State1 to State2A or State2B but the trigger should only run when the object is promoted to State2A. This attribute should contain state_State2A.

3. If this is the first trigger program of this kind for the event (the Trigger Manager program and input has not been specified for this kind of trigger for this event), proceed to step 4.

Or:

If there is already a trigger program of this kind for the event:

- Make sure the sequence numbers for the existing trigger programs are correct. If the program you added should run before the existing programs, the sequences for the existing programs will probably have to change. For instructions on changing the sequence number for a trigger program, see [Modifying Inputs Trigger Programs](#).
- The name of the Trigger Program Parameter should already have been specified as the input for the Trigger Manager. Therefore, the procedure is complete and you can skip the remaining steps.

4. Promote the object to the Active state.

5. In Business Modeler, find the administrative object for which you want to add the trigger.

For example, to have a program run when a part is promoted to the Release state, find the policy that governs the part. To have a program run when a file is checked into an object, find that object type.

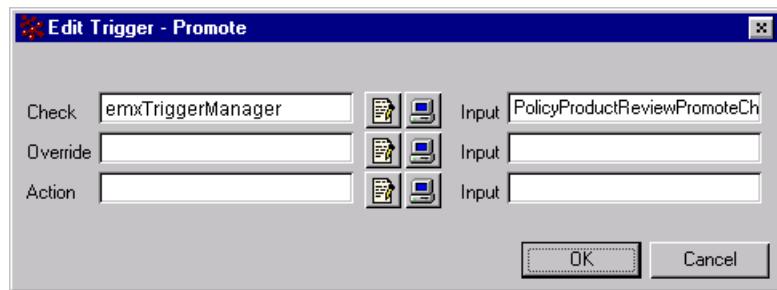
6. Double-click the administrative object to edit it.

7. Select the **Trigger** tab. (For policies, double-click the appropriate state and then select the **Trigger** tab.)

8. In the box for the type of trigger program that you want to add (Check, Override, Action), enter **emxTriggerManager**.

9. In the **Input** box to the right of the program name, enter the name of the eService Trigger Program Parameters object that represents the trigger program you want to run. Make sure the name is *exactly* the same as the Trigger Program Parameters object.

If there are eService Trigger Program Parameters objects with different names that need to be called--for example, **TypeAllCreateAction** and **TypePartCreateAction**--you can enter both names separated by a space.



Validate Query Trigger

You can add a trigger to the query execution event so that administrators may control the types of queries that users are allowed to perform. You can customize the behavior of queries (even block them from evaluating) depending on the query patterns specified.

The following topics are discussed:

- [About the Validate Query Trigger](#)
- [Enabling the Validate Query Trigger](#)
- [Writing the ValidateQuery Program](#)
- [Validate Query Use Case](#)

About the Validate Query Trigger

This section provides background details about the validate query trigger.

The trigger is fired from the Find or Find Like windows in both the desktop and Web versions of Matrix Navigator, where wide-open query definitions are allowed, as well as on specific queries in MQL or Studio Customization Toolkit programs. The Validate Query trigger has two main purposes:

- To block certain types of queries that are inefficient
- To change the query patterns in the background to perform a more specific query than originally attempted. For example, if the vault pattern is a wildcard (*), the value can be changed to be a list of all local vaults instead.

In addition, you can create customized error and notice messages. For example, for a blocked query, the program could display a message indicating the reason for blocking it, such as more information is required.

The validate query trigger is different from other triggers. It executes a single program that works as a combined check and override trigger, and is designed as a way to modify the current query (via the RPE).



Enabling the Validate Query Trigger

The validate query trigger is enabled in the desktop application by default, as long as a program object named ValidateQuery exists in the database.

Because it makes an additional call to the database, performance can be affected. To enable Query triggers for use in the Web version of Matrix Navigator, add the following to the HTML page that starts it:

```
document.writeln("<param name='QueryTrigger' value='True'>");
```

If you will use custom applets, your pages may use the straight HTML syntax:

```
<PARAM name="QueryTrigger" value="True">
```

In MQL, the `evaluate query` and `temp query` commands can include the `querytrigger` clause, which will cause the ValidateQuery program execute (if it exists) For example:

```
<mql> temp query bus * * * querytrigger vault v1;
<mql> evaluate query ql querytrigger;
```

Even with triggers turned off, when `querytrigger` is included in a query command, the ValidateQuery program executes.

Program objects that perform queries are not affected by default in either the desktop or Web version of Matrix Navigator. If required, the `querytrigger` flag must be included in the program code.

Using the Validate Query Trigger in Studio Customization Toolkit Program

Studio Customization Toolkit methods are available to set and get the query trigger flag.

By default the flag is set to false. The methods are:

```
public setQueryTrigger(boolean queryTrigger)
public boolean getQueryTrigger ()
```

The following example performs a temporary query and the validate query trigger fires as long as the program object exists:

```
Query myQuery = new Query("");
myQuery.setType("*");
myQuery.setName("*");
myQuery.setRevision("*");
myQuery.setVault("*");
myQuery.setQueryTrigger(true);
myQuery.evaluate(getContext());
...
```



Writing the ValidateQuery Program

The validate query trigger fires before the query is evaluated.

The program should check the query patterns specified as described in [Query Event Macros](#).

If the ValidateQuery program is a JPO, no macros are available as input arguments. As a workaround, the ValidateQuery program should be a Tcl program that calls the JPO to do the actual work.

When changing query patterns, use global variables. For example, to change the vault pattern in the background if only a wildcard is provided, you could use:

```
tcl;
eval {
    set sVault [mql get env VAULT_PATTERN ]
    if { $sVault=="*" } {
        set sVault "vault1,vault2,vault3"
    }
    mql set env global VAULT_PATTERN
exit 0
}
```

As is the case with other event triggers, the exit code determines if the event is blocked or overridden. If the exit code is a zero, the query is performed: the global query pattern RPE variables are read and used by the query, which allows the program code to change them before the query is evaluated. (Remember, the validate query trigger is really a combination of a check and an override trigger.) If the exit code is a non-zero value, a soft block occurs. In this case the query as entered is not executed; the query event is overridden and the transaction is committed, closing the Find or Find Like window. It is up to the trigger program code that overrides the query to generate a message to inform the user that the query did not get executed.

While overriding the query code with other events is possible, the main purpose of a query trigger is to block unqualified queries and modify the query patterns via RPE variables. If overriding the query code, query performance, Tcl limitations (such as the fact that Tcl is single threaded) and recursive queries must be considered. The query is a fundamental part of ENOVIA Live Collaboration and overwriting the code must be done carefully. To return business objects to the current window in the desktop or Web version of Matrix Navigator, the MQL commands, `appl icons` and `appl details` can be used to populate the window with business objects.



Validate Query Use Case

The validate query trigger can ensure that user-defined queries satisfy some pre-defined requirements.

For example, the program below is one solution for ensuring that:

- All queries in local vaults have at least one filter (for example, type, name, rev, or owner for basic queries), but partial wildcards are permitted.
- A query on all vaults (*) does not include foreign vaults.
- Queries on a specified foreign vault have both type and name filters.

The ValidateQuery program to satisfy above:

```
tcl;
eval {
    eval [ mql print program utList.tcl select code dump ]
#####
proc tclListToMxList {tclList} {

    set mxList ""

    foreach i $tclList {
        append mxList $i
        append mxList ","
    }
    set length [expr [string length $mxList] - 2 ]
    set mxList [ string range $mxList 0 $length ]
    return $mxList
}

#####
proc mxListToTclList {mxList} {
```

```

        set tclList [ split $mxList "," ]
        return $tclList
    }

#####
#proc MatchingVaults {sPattern tclVaultList} {

    set lVaultsToBeSearched ""
    set firstIndex [ lsearch $tclVaultList $sPattern ]
    while { $firstIndex != -1 } {
        lappend lVaultsToBeSearched [ lindex $tclVaultList
$firstIndex]
        set index [expr $firstIndex + 1 ]
        set tclVaultList [lrange $tclVaultList $index end]
        set firstIndex [ lsearch $tclVaultList $sPattern ]
    }
    return $lVaultsToBeSearched
}

#####

#proc onlyWildcardsInString {sString} {

    set wildcard "*"
    set bReturn FALSE
    set iFirstIndex [string first $wildcard $sString]
    if {$iFirstIndex != -1 } {

        # To discover many stars (ex ***)
        set bReturn TRUE
        for {set i 0} {$i < [string length $sString]} {incr i}
        {
            if { [string index $sString $i] != $wildcard } {
                set bReturn FALSE
                break
            }
        }
        return $bReturn
    }

#####

#
# Variable definition
#
#####

# Default values
set iExit 0
set sErrorMessage ""

# Get all vaults from Matrix
set sAllVaultsList [ mql list vault ]
set lAllVaultsList [ split $sAllVaultsList \n ]

# Defined adaplets
set lAdapletVaults {a1 a2}

```

```

set lRemoteVaults {r1 r2}

# Local vaults
set lLocalVaults [ utListSubtract $lAllVaultsList [ concat
$lAdapletVaults $lRemoteVaults ] ]

# Get query env
set sVault [mql get env VAULT_PATTERN ]
set sType [mql get env TYPE_PATTERN ]
set sName [mql get env NAME_PATTERN ]
set sRevision [mql get env REVISION_PATTERN ]
set sOwner [mql get env OWNER_PATTERN ]

set lVaultsToBeSearched {}

#####
#
# Find out which vault(s)are going to be queried
#
#####

# If vault pattern is a wildcard change the vault to only
search local vaults
if { [onlyWildcardsInString $sVault] } {
    set sVault [ tclListToMxList $lLocalVaults ]
}
mql set env global VAULT_PATTERN $sVault

# if vault pattern is a comma separated list of vaults
if { [string match *,* $sVault] == 1} {
    set lVaultList [ split $sVault "," ]
    foreach v $lVaultList {
        set lVaultsToBeSearched [ concat $lVaultsToBeSearched
[MatchingVaults $v $lAllVaultsList ] ]
    }
}
# not a list of vaults
} else {
    set lVaultsToBeSearched [MatchingVaults $sVault
$lAllVaultsList ]
}

#####
#
# Go through the vaults that are going to be searched and
# check if all criteria are satisfied
#
#####

set bGeneralCheck FALSE

foreach v $lVaultsToBeSearched {
    # Foreign vault al is not allowed to have * only ? is
allowed
    if { $v == "al" } {
        if { ([string first * $sType] != -1) && ([string first
* $sName] != -1) } {

```

```

        set iExit 1
        append sErrorMessage "Query not allowed. When
searching in Vault $v, none of the query patterns type and name
is allowed include a wildcard (*). Use one or many question
marks (?) for unknown characters instead\n\n"
    }
} elseif {$v == "a2"} {
    if { [string first * $sName] != -1 } {
        set iExit 1
        append sErrorMessage "Query not allowed. When
searching in Vault $v, the query patterns type is allowed
include a wildcard (*). Use one or many question marks (?) for
unknown characters instead\n\n"
    }
} else {
    set bGeneralCheck TRUE
}
}

if {$bGeneralCheck} {
    set iAllowedWildards 3
    set iCurrent 0

    if { [onlyWildcardsInString $sType] } {
        incr iCurrent
    }
    if { [onlyWildcardsInString $sName] } {
        incr iCurrent
    }
    if { [onlyWildcardsInString $sRevision] } {
        incr iCurrent
    }
    if { [onlyWildcardsInString $sOwner] } {
        incr iCurrent
    }
    if {$iCurrent > $iAllowedWildards } {
        set iExit 1
        append sErrorMessage "Query not allowed. One of the
query patterns type, name, revision or owner must be more
qualified than a wildcard (*)\n\n"
    }
}

#
# Pop up error message if needed
#
if {$iExit == 1} {
    mql error $sErrorMessage
}

exit $iExit
}

```

Recursion Detection Modes and Limits

This section discusses the use of recursion with triggers.

The following topics are discussed:

- [Types of Recursion](#)
- [Trigger Recursion Scenarios](#)
- [Limiting the Trigger Stack Depth](#)
- [Override Example](#)

Types of Recursion

ENOVIA products allow program objects to execute themselves, a process called direct or explicit recursion.

Explicit recursion is a powerful mechanism when used properly. Event triggers allow indirect or implicit recursion that occurs when a trigger program generates the same event with which it is associated. This form of recursion may or may not be useful. Implicit recursion might be difficult to determine because the event that triggers the running program may be part of a program that was triggered by a totally different event. The amount of indirection could be many levels deep, and difficult to figure out.



Trigger Recursion Scenarios

You can configure how your system detects trigger recursion by setting the MX_TRIGGER_RECUSION_DETECTION environment variable.

There are 3 scenarios:

1. By default, this variable is set to `name`, and all recursion encountered through event triggers is treated as a no-op (that is, if an attempt is made to execute a running trigger program, the attempt fails but is not treated as an error). The system stores the name of the executing program, and uses a name matching test to determine if recursion is being attempted. If a case of recursion is encountered, the system simply skips execution of that trigger program but otherwise continues to process in non-error mode.
2. Since programs can be configured with input arguments, it is reasonable to design programs for reuse, and therefore, it may be desirable to allow running programs to be executed again. For example, you may have a program that promotes an object that it is passed. When this program is executed, it could fire a promote trigger that calls the same program, but passes in a different object. To allow this, you could use a signature matching test by setting the variable to `signature`. When signature matching is enabled, the system stores the following five properties of a trigger program when it is executed:
 - Program Name
 - Input Arguments
 - Trigger Event
 - Trigger Type
 - Target object ID

The target object ID is a unique internal number that identifies the business object or relationship instance that is the target of the given event. When the event is created on a business object or on a relationship instance, a unique number is generated but this number does not identify an object (since one does not yet exist). This means that there is a much higher potential for unwanted recursion in these two cases so special attention should be paid when writing trigger programs.

If a program attempts to execute where all five of these properties match another program on the running stack, recursion is detected and the routine is skipped.

3. When full recursion is desired, the detection of recursion can be turned off by setting the variable to `none`. In this case, it is important to avoid infinite recursion using the environment variable MX_TRIGGER_RECUSION_LIMIT.



Limiting the Trigger Stack Depth

The MX_TRIGGER_RECUSION_LIMIT variable sets the number of trigger programs that may be placed on the runtime stack to the value set, helping to avoid server crashes due to the host system running out of stack space. The default value is 30 (which may be too high, particularly when using the Web version of Matrix Navigator through a Windows server).

When the limit is reached, a warning is added to the trigger trace file such as:

```
"WARNING: Override Trigger for <event>: <program_name> has  
exceeded recursion limit!"  
"WARNING: Validate Query for <event>: <program_name> has  
exceeded recursion limit!"
```

The limit is used in conjunction with MX_TRIGGER_RECUSION_DETECTION, no matter what type of recursion detection is enabled.

Override Example

The following example shows what happens when trigger programs issue commands that cause events to occur that also have triggers.

This example involves the business object "Document Checker 0" and a bit of a squabble between three users, Tom, Dick, and Harry, who fight over ownership. The default recursion test (name matching) is in use.

Assume that these triggers are in place in the Document type on the changeowner event:

```
# Changeowner Event Check Program on Document Type:
mql
code "output 'DocumentChangeOwnerCheck: OBJECT = ${OBJECT} TYPE
= ${TYPE}'";
output ' NAME = ${NAME} REVISION = ${REVISION} OWNER =
${OWNER}';
output ' ACCESSFLAG = ${ACCESSFLAG} USER = ${USER}';
output ' CHECKACCESSFLAG = ${CHECKACCESSFLAG}';
output ' TIMESTAMP = ${TIMESTAMP}';
output ' NEWOWNER = ${NEWOWNER}';
tcl;
set status 0
set status $env(RETURN_CHECK_CODE)
exit $status
";
#Changeowner event Override Program on Document #Type:
mql
code "output 'DocumentChangeOwnerOverride: OBJECT = ${OBJECT}
TYPE =${TYPE}';
output ' NAME = ${NAME} REVISION = ${REVISION} OWNER =
${OWNER}';
output ' ACCESSFLAG = ${ACCESSFLAG} USER = ${USER}';
output ' CHECKACCESSFLAG = ${CHECKACCESSFLAG}';
output ' TIMESTAMP = ${TIMESTAMP}';
output ' NEWOWNER = ${NEWOWNER}';
modify bus ${TYPE} ${NAME} ${REVISION} owner Dick;
tcl;
set status 0
set status $env(RETURN_OVERRIDE_CODE)
exit $status
";
```

Environment variables handle the exit code from the check and override trigger programs. Their settings are:

`RETURN_CHECK_CODE=0` and `RETURN_OVERRIDE_CODE=1` (which means the check program will not block and the override program will replace the normal event handling). Harry is the current owner of the Document object and has decided to change ownership to Tom. Here is the output:

```
MQL<7>modify bus Document Checker 0 owner Tom;
DocumentChangeOwnerCheck: OBJECT = Document Checker 0 TYPE =
Document
    NAME = Checker REVISION = 0 OWNER = Harry
    ACCESSFLAG = True USER = creator
    CHECKACCESSFLAG = True
    TIMESTAMP = Thu Jan 6, 2010 11:28:03 AM
    NEWOWNER = Tom
DocumentChangeOwnerOverride: OBJECT = Document Checker 0 TYPE =
Document
    NAME = Checker REVISION = 0 OWNER = Harry
    ACCESSFLAG = True USER = creator
    CHECKACCESSFLAG = True
    TIMESTAMP = Thu Jan 6, 2011 11:28:04 AM
    NEWOWNER = Tom
DocumentChangeOwnerCheck: OBJECT = Document Checker 0 TYPE =
Document
    NAME = Checker REVISION = 0 OWNER = Harry
    ACCESSFLAG = True USER = creator
    CHECKACCESSFLAG = True
    TIMESTAMP = Thu Jan 6, 2011 11:28:05 AM
    NEWOWNER = Dick
```

The changeowner override program issues its own `modify businessobject Document EventTriggers 0 owner Dick`

command. When Harry attempts to change the owner, this command causes a nested changeowner event to occur, which causes a second set of triggers to fire. The second override program is a no-op because the first override program is still running. Since this second override program is a no-op, the normal event activity (in this case, the first override program) is guaranteed to take place. This is an important point to keep in mind. At this point, the owner has been changed to Dick.

After the nested changeowner event transaction commits and the associated action program, if any, runs, control returns to the original changeowner event. If the exit code from the override program is non-zero, the normal event activity is skipped and the owner name continues to be Dick. If the exit code is zero, the normal activity takes place (overriding the owner change that took place in the nested event transaction). Now the owner name is changed to Tom. In either case, the original event transaction commits and the action program (if any) executes.

Look at the case of the changeowner action program issuing its own `modify businessobject Document Checker 0 owner Harry` command. Assume that `RETURN_OVERRIDE_CODE=0` so that the event is not replaced.

#Changeowner event Action Program on Document Type

```
mql
code "output 'DocumentChangeOwnerAction: OBJECT = ${OBJECT} TYPE
=${TYPE}';
output ' NAME = ${NAME} REVISION = ${REVISION} OWNER =
${OWNER}';
output ' USER = ${USER}';
output ' TIMESTAMP = ${TIMESTAMP}';
output ' NEWOWNER = ${NEWOWNER}';
modify bus ${TYPE} ${NAME} ${REVISION} owner Harry;
quit;"
```

In this case, the command causes a new changeowner event to occur after the initial event transaction has been committed. When it is time for the action program to be run, the owner has changed to Tom. As before, recursion occurs and the second action program is a no-op since the first action program is still running. Note that the second owner change to Harry, which is caused by the action program, will override the owner change to Tom that took place as the normal event transaction before the action program was executed.

Look what happens when both the changeowner override and action programs generate changeowner events as defined earlier. Assume that `RETURN_OVERRIDE_CODE=1` again, so that the event is replaced. In this case, we see the combined affects of both programs. The question is who now owns the Document object? The only way to find out is to trace through the changes in a sequential fashion.

```
1.0) Original ChangeOwner event transaction is started. Check fires.
2.0) Override changes owner to Dick.
    2.1) Nested ChangeOwner transaction is started. Check fires.
    2.2) Override is a no-op, since the same Override Program (step 2.0) is still
running. Transaction is committed.
    2.3) Nested event's Action changes owner to Harry.
        2.3.1) Nested ChangeOwner transaction is started. Check fires.
        2.3.2) Override is a no-op, since the same Override Program (step 2.0) is still
running. Transaction is committed.
    2.4) Nested Action is a no-op, since the Action (step 2.3) is still running.
Transaction is committed.
3.0) Original ChangeOwner event Action changes owner to Harry. (redundant).
```

With the original event transaction now committed, it seems as if the action program changes will prevail.

```
3.1) ChangeOwner transaction is started. Check fires.
3.2) Override changes the owner to Dick.
    3.2.1) Nested Changeowner transaction is started. Check fires.
    3.2.2) Override is a no-op since the Override program (step 3.2) is still
running. Transaction is committed.
    3.3.3) Nested Action is a no-op since the Action program (step 3.0) is still
running. Transaction is committed.
    3.3) Action is a no-op since the Action program (step 3.0) is still running.
Transaction is committed.
```

The proper conclusion is that an override program that replaces normal event activity will always win out. This example is simplistic because the check program was not involved, and the override and action programs issued single commands that generated the same event. Obviously, trigger programs issue many commands and the events generated by these commands trigger other programs, and so on. Because of this complexity, consider graphing the events, showing time on one axis, and the level of nesting on the other.

Enabling/Disabling Triggers

Triggers can be disabled for a session without having to explicitly remove them from the administrative definitions to which they belong. This is helpful during testing and troubleshooting.

You can execute the following command from MQL provided that the current user is defined as a System Administrator.

```
trigger off;
```

After this command executes, events that occur on the local machine which have associated triggers do not cause the programs to execute.

This command affects only the local machine; concurrent user sessions are not affected. Even when multiple users are sharing one executable, each user is isolated from all other users. In addition, subsequent sessions on the local machine will have triggers enabled by default.

To enable defined triggers again, a System Administrator should use:

```
trigger on;
```

MQL also allows triggers to be enabled or disabled with the use of a toggle command. Depending on the current setting, triggers can be enabled/disabled using the same command.

To enable/disable triggers as a toggle, a System Administrator should use:

```
trigger;
```

Utility Trigger Programs Reference

This section describes the utility trigger programs used by ENOVIA products installed with Business Process Services. You specify input values in the same way for all trigger programs.

In this section:

- [Previous Revision Promotion](#)
- [Required Connection Check](#)
- [Required File Check](#)
- [Check Relative State](#)
- [Valid Revision Sequence](#)
- [Relative Float Action](#)
- [Set Originator Attribute](#)

Previous Revision Promotion

eServicecommonPreviousRevisionPromotion_if.tcl is an action trigger for a promote event.

The program promotes to a specified state the previous revision of the object being promoted. This ensures that when an object is promoted to a particular state, its previous revision is also promoted. If the program encounters a trigger attached to the state to which the revision is getting promoted, the program displays an error message.

The following topics are discussed:

- [Arguments](#)
- [Returns](#)

Arguments

Argument Number	Argument Name	Description and Inputs
1	sState	<p>The state to promote the previous revision. The program gets the state names from the policy that governs the object being promoted. The valid values for this argument include:</p> <p>next--Promotes the previous revision to the next state regardless of the current state of the object being promoted.</p> <p>last--Promotes the previous revision to the last state in the lifecycle.</p> <p>[specific state]--Promotes the previous revision to the specified state.</p> <p>If you pass null as the input, the system promotes the previous revision to the current state of the object being promoted.</p>



Returns

0 if the previous revision is successfully promoted; otherwise 1 and an error message.

Required Connection Check

eServicecommonTrigcRequiredConnection_if.tcl checks to see if an object has specific connections.

The following topics are discussed:

- [Information to Check](#)
- [Arguments](#)
- [Returns](#)

Information to Check

The following table shows the kinds of information the program can check for.

Does the selected object have any:	For example:
Connections with other objects.	Does the object have any relationships?
Connections with other objects in one direction.	Does the object have any relationships in which the object is on the From end?
Connections with other objects through specified relationships.	Does the object have any connections to other objects with the Includes Part relationship.
Relationships with objects of specified types.	Does the object have any connections to a Part object?

You can combine the checks listed above. For example, you can have the program check if the object has a connection in which the object is on the From end of any Supplier Response relationship and in which the connected object is an Initial Quotation type.



Arguments

Argument Number	Argument Name	Description and Inputs
1	Direction	<p>The direction(s) the program should search from the selected object to connected objects. The program accepts three values:</p> <ul style="list-style-type: none">• from--Search only connections where the selected object is on the From side of the relationship. <p>For example, suppose three objects are connected like this:</p> <p>Object 1 --> Object 2 --> Object 3</p> <p>If the direction input is "from" and Object 2 is the selected object, then the program only considers the connection between Objects 2 and 3.</p> <ul style="list-style-type: none">• to--Search only connections where the selected object is on the to side of the relationship.• ""(blank)--Search all connections regardless of the direction.
2	RelationList	<p>The relationships the program should check for, formatted as a comma-delimited list of symbolic names for the relationships. For example, to check for the Includes Part and Supplier Response relationships, the input would be: <code>relationship_IncludesPart,relationship_SupplierResponse</code></p>
3	TypeList	<p>The types the program should check for, formatted as a comma-delimited list of symbolic names for the types. For example, to check for connections with Initial Quotation and Final Quotation objects, the input would be: <code>type_InitialQuotation,type_FinalQuotation</code></p>

This table shows how to configure the argument inputs depending on the kind of check you want to perform.

To check for:	Use these inputs:		RelationList	TypeList
	Direction			
Any connection	blank		blank	blank
			blank	blank

Any connection in specified direction	To or From		
Any connection to objects of specified type	To, From, blank	blank	specified
Any connection of specified relationship	To, From, blank	specified	blank
Any connection of specified relationship to objects of specified type	To, From, blank	specified	specified



Returns

0 if at least one specified connection exists. Otherwise the program returns 1 and an error message indicating no connection was found and giving details of the specified connection.

Required File Check

eServicecommonTrigcRequiredFormat_if.tcl checks to see if any files are checked in to a selected object.

File could be checked in to:

- Selected object
- Selected object for the default format
- Selected object for a specified format

The following topics are discussed:

- [Arguments](#)
- [Processing](#)
- [Returns](#)

Arguments

Argument Number	Argument Name	Description and Inputs
1	sSearchKey	Specifies the format(s) to check. The following values are valid for this argument: list--Check formats listed in IFormatPropertyList argument. default--Check default format of selected object. all--Check all formats supported by selected object.
2	IFormatPropertyList	This argument has meaning only if sSearchKey is set to "list". The argument accepts a list of the symbolic names of the formats to check. For example, {format_Word format_generic ?}.
3	bSizeCheck	Checks whether the size of checked in files is 0 bytes. The argument accepts two values: True--Checks the size of all the checked files (of the format specified by the sSearchKey argument) and fails if all the checked in files are 0 bytes. False--Does not check the file sizes.



Processing

If the input for sSearchKey is:	The program:
list	Searches for checked files in all the formats listed in argument IFormatPropertyList. If any format is not supported by the selected object, the program ignores that format.
default	Searches for checked in files in only the default format for the selected object.
all	Searches for checked in files in all the supported formats of selected object.



Returns

If bSizeCheck is False and:

- The program finds a checked in file of the specified format, it returns 0 for success.
- The program does not find a checked in file of the specified format, it returns 1 along with an MQL message that explains the reason for failure.

If bSizeCheck is True and:

- The program finds a checked in file of the specified format and the checked in files are not 0 bytes, it returns 0 for success.
- The program does not find a checked in file of the specified format or the checked in files are all 0 bytes, it returns 1 along with an MQL message that explains the reason for failure.

Check Relative State

eServicecommonCheckRelState_if.tcl is a check trigger for a promote event. It searches a specified relationship looking for a specified object type and determines if all of the objects found are at the minimum acceptable state specified.

The following topics are discussed:

- [Arguments](#)
- [Returns](#)

Arguments

Argument Number	Argument Name	Description and Inputs
1	sRelationship	The relationship type to search for. If the argument input is null, the program checks all the relationships in the direction specified for sDirection. If the specified relationship is not assigned to the target object type, the program returns a FAIL.
2	sTargetObject	The object type to search for at the other end of the relationship. If null, the program checks all the objects that can be reached through the relationship specified for sRelationship.
3	sTargetState	The state the target objects should be in. If the specified state is not a valid state defined for the target object, then the program returns a FAIL.
4	sDirection	The direction to search for. The program accepts three values: TO, FROM, BOTH. If null, the program searches in the FROM direction.
5	sComparisonOperator	The operator the program should use for checking the target state. The values are: LT (Less Than)--The program returns a FAIL if the target object is in a state greater than the target state. GT (Greater Than)--The program returns a FAIL if the target object is in a state less than the specified state. EQ (Equal)--The program returns a FAIL if the target object is in a state other than the specified state. LE (Less Than Or Equal)--The program returns a FAIL if the target object is in a state greater than or equal to the specified state. GE (Greater Than Or Equal)--The program returns a FAIL if the target object is in a state less than or equal to the specified state. NE (Not Equal)--The program returns a FAIL if the target object is in the specified state. If null, the program uses EQ.
6	sObjectRequirement	A flag that specifies whether the object is required or optional. The values are: Required and Optional. If null, the program uses Optional.



Returns

Pass if all objects are in the minimum required state. Otherwise, the program returns fail. The specific conditions for passing and failing depend on the Comparison Operator. For example, suppose the comparison operator is LT (less than) and the target object has states defined as A, B, and C.

If the target state is:	And the current state is:	The program returns:
A	A, B, C	fail
B	A	pass
B	B, C	fail
C	A, B	pass
C	C	fail

Now suppose the comparison operator is EQ (equals).

If the target state is:	And the current state is:	The program returns:
A	A	pass
A	B, C	fail
B	A, C	fail
B	B	pass
C	C	pass
C	A, B	fail

Valid Revision Sequence

eServiceValidRevisionChange_if.tcl is a check trigger for a create event. This program ensures the revision sequence is maintained for a business object by not allowing an object to be created if there is an existing object of the same type and name and of any revision.

The following topics are discussed:

- [Arguments](#)
- [Returns](#)

Arguments

The program accepts no arguments.



Returns

If the program finds no object of the same type and name (with any revision), it returns zero and the object is created. If the program finds an object of the same type and name with any revision, the program returns a non-zero value, prevents the create event, and shows a message telling the user to use **Object > New > Revision** instead of **Object > New > Original**.

Relative Float Action

The program eServicecommonRelativeFloatAction_if.tcl is a promote action trigger. When an object is promoted, the program gets the previous revision of the object and floats all the specified relationships in the specified direction.

The following topics are discussed:

- [Arguments](#)
- [Processing](#)

Arguments

Argument Number	Argument Name	Description and Inputs
1	sRel	The relationship to float. Must be the symbolic name (for example, relationship_PartRule).
2	sDirection	The relationship direction to float. Accepts three values: from, to and both.
3	sOperation	A flag that specifies which revision of the object to look for. It accepts two values: PREVIOUS, LATEST
4	sState	If argument 3 is LATEST, this argument should specify the state in which the latest revision should be. It is passed in as the symbolic name for a state (for example, state_Release).



Processing

Suppose there are four versions for an object A: A1, A2, A3 and A4. The trigger gets fired when A4 is promoted to its next state. If the third argument input is LATEST, the trigger program finds the LATEST revision for A that is in the state specified by the fourth argument. If the third argument is PREVIOUS, the program gets the previous revision, which is A3. It then traverses through the relationship specified in argument 1 in the direction specified by argument 2 and gets all the connected objects, then connects them to A4.

Set Originator Attribute

The Java Program Object emxcommonSetOriginator_if is an action program for create and revise events. The program sets the Originator attribute value to the name of the current user. The program looks up the actual Originator attribute name because the customer could change name of the attribute.

When the trigger program runs, it retrieves the necessary environment variables, such as TYPE, NAME, REVISION, VAULT, USER, EVENT, NEWREV, and APPREALUSER. Because creation is often done as the Shadow Agent person, the program checks the environment variable APPREALUSER. If set, it is used instead of USER for setting the originator.

Two eService Trigger Program Parameters objects that call this trigger program are installed with Business Process Services. These objects call the program whenever any type is created or modified. See "Automation for All Types" in the *Live Collaboration Administrator's Guide*.

The following topics are discussed:

- [Arguments](#)
- [Returns](#)

Arguments

Argument Number	Argument Name	Description and Inputs
1	attrName	The symbolic name for the attribute that is to be populated.



Returns

0 for success; non-zero for failure.

Charts

This section describes how to setup the components necessary to use charts within an application. The charting software is distributed with Business Process Services and is automatically available.

In this section:

- [Working With Charts](#)
- [Adding a Chart to a Page](#)
- [URL Parameters for emxChart.jsp](#)

Working With Charts

The chart component draws different types of charts such as bar, stacked bar, pie and line charts based on table numerical column values.

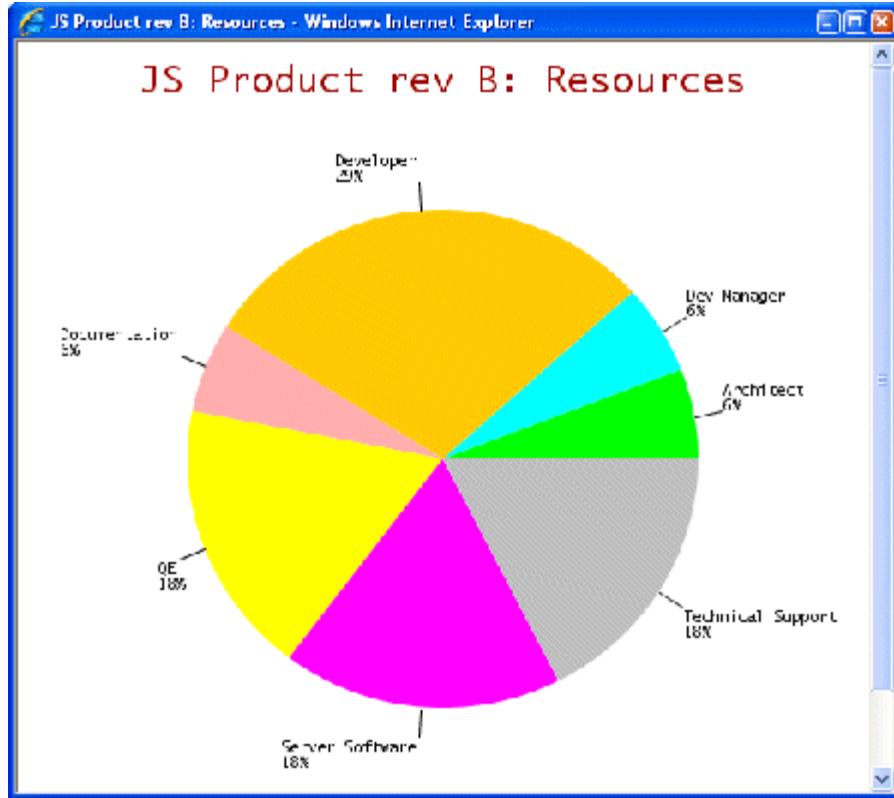
To create charts, ENOVIA products use KavaChart, which is installed with the Business Process Services software. For information about KavaChart, see <http://www.ve.com>.

You can use charts anywhere, for example:

- A tab in a portal channel
- A node of a tree
- An HREF in any page, for example, as a custom link in a table

The following figure shows a sample chart in a portal channel.

- If you add the chart page to a portal channel, then by default the toolbar is shown with the launch icon in the toolbar. The launch icon can then be used to open a channel page in a larger window.
- The toolbar does not display if the chart page is in a node of a tree or a HREF in any page.



Adding a Chart to a Page

The chart component uses `emxChart.jsp` to draw a chart in the page using the URL parameters passed to it. You can display a chart within a page, or from within a configurable table.

1. To display a chart directly within a page, call `emxChart.jsp` with the appropriate URL parameters.

For example:

```
emxChart.jsp?program=emxCharacteristic:getCharacteristicObjects  
&table=ENGFurtherChemicalPhysicalCharacteristicList&header  
=emxEngineeringCentral.Common.Further_Chemical_Physical_Characteristic.Heading  
&type=type_FurtherChemicalPhysicalCharacteristic&chartType=PieChart&&XAxis=Name&YAxis=Value
```

See [URL Parameters for emxChart.jsp](#) for the list of allowed URL parameters.

2. To display a chart from an existing configurable table page, define at least one column in the chart as numeric.

A column is numeric if one of the following conditions is satisfied:

- You configure the column to show a numeric attribute type (attribute of type `integer` or `real`) values. If this numeric column has a setting `format` assigned to `string`, then this column is considered as non-numerical column.
- You configure the column to show string values (by defining an attribute expression or any valid businessobject/relationship expression or a program), with the setting `format = numeric`.

If the attribute displayed in a column is defined with a dimension, then the values are compared or grouped using the normalized values. The chart labels will show the normalized unit of measure.

An icon appears in the table toolbar in view mode whenever a table has numeric columns. When clicked, it shows the Chart Options dialog where users select various options on any given table instance and then draw the chart based on the selections. For details on the Chart Options dialog, see the *Application Exchange Framework User's Guide*.

URL Parameters for emxChart.jsp

This table lists the parameters that you can use with emxChart.jsp. You can add these parameters to the href parameter for the component that calls the chart.

If the ENOVIA Live Collaboration Server is installed on a UNIX or Linux system that does not have the X-Windows server installed, for example a system that does not have a display, you will need X-Windows if you want to use charts in an application. For details, see "Using a UNIX ENOVIA Live Collaboration Server without a Monitor" in the *ENOVIA Live Collaboration Installation Guide*.

Parameter	Description	Accepted Input Values
table	Assigned to a table administrative object to be used for presenting this chart page.	table=SCSBuyerDesk table=ENCParts
chartType	The type of the chart to be drawn.	BarChart StackBarChart PieChart LineChart
chartTitle	The title for the chart.	
draw3D	Indicates whether the chart is drawn in 3D or not.	true--Draws the chart in 3D false--Draws the chart in normal mode (default)
header	Header to display on top of the chart.	emxFramework.Chart.header
inquiry	Assigned to one inquiry administrative object name. The inquiry will be used to get the objects used to draw the chart.	inquiry=SCSBuyerDesk inquiry=ENCAIIIParts
program	<p>Used only when there is no <code>inquiry</code> parameter passed to emxChart.jsp.</p> <p>This parameter is used as an alternative approach to the <code>inquiry</code> object approach. This <code>program</code> approach uses the JPO program object to get the object list to be displayed in the table.</p> <p>This parameter is assigned to one set of values that will form a JPO program name and the method name. The format of the parameter value is:</p> <pre>program=<JPO program name>:<JPO method name></pre> <p>The program name and the method name are separated by a colon ":".</p> <p>The parameter value may have one or more sets of values separated by comma ",".</p> <p>The first set of values (JPO name and method name) are used to get the object list when loading first. All programs are associated with the table filter in the header.</p> <p>If there is only one program value, the filter in the header will not be shown.</p>	program=emxTableBuyerDesk:getBuyerDesk
XAxis	<p>Table column name to be used as the x-axis for bar charts or stacked bar charts.</p> <p>Table column name to be used as the "slice" label for pie charts.</p> <p>Table column name to be used as the line label for line charts.</p> <p>This should be a non-numeric column except for line charts.</p>	Count
YAxis	Comma-separated list of numeric column names to be used as the y-axis for bar charts, stacked bar charts, or	totalCount weight

	<p>line charts.</p> <p>Table column name to be used as the pie data for pie charts.</p>	
labelDirection	<p>Used for bar charts, stacked bar charts, and line charts to specify whether to display the x-axis in a horizontal or vertical direction.</p>	<p>horizontal - Draws the x-axis label horizontally (default).</p> <p>vertical - Draws the x-axis label vertically.</p>

Reports

All ENOVIA products include support for Crystal Reports. To use Crystal Reports, you must write custom reports.

In this section:

- [About Reports](#)
- [Installing BusinessObjects XI Release 2](#)
- [Installing Crystal Reports Professional](#)
- [Verifying the Configuration](#)
- [Converting Crystal Reports 10](#)
- [Running Sample Reports](#)
- [Modifying the System Properties for Reports](#)
- [Implementing the Crystal Reports Interface](#)
- [Creating a Crystal Report File with the MatrixJPO Data Source](#)
- [Creating a JPO for the RPT File](#)
- [Creating a Crystal Report File with the MatrixInquiry Data Source](#)
- [Creating an Inquiry To Be Used by the RPT File](#)
- [Sample Reports and JPOs](#)

About Reports

All ENOVIA products include support for Crystal Reports. Crystal Reports is integrated so that you can run a preconfigured report from within the JSP environment.

The following topics are discussed:

- [Software Components You Need to Install](#)
- [Software Requirements](#)
- [Core Support](#)

Software Components You Need to Install

You must install these two software components:

- BusinessObjects XI Release 2: a Web-based report management and delivery system that can be used to navigate, print, and export reports
- Crystal Reports: a thick-client report designer used for creating your own reports that use data from the ENOVIA Live Collaboration database



Software Requirements

To use Crystal Reports, your installation must meet these requirements.

- ENOVIA Live Collaboration Server version 10.7 or higher
- ENOVIA-supported application server
- ENOVIA-supported browser
- Studio Modeling Platform version 10.7 or higher
- Business Process Services version 10.7 or higher
- BusinessObjects XI Release 2
- Crystal Reports XI R2 Professional
- Java 1.5
- Microsoft IIS on Windows Web server or Apache TomCat



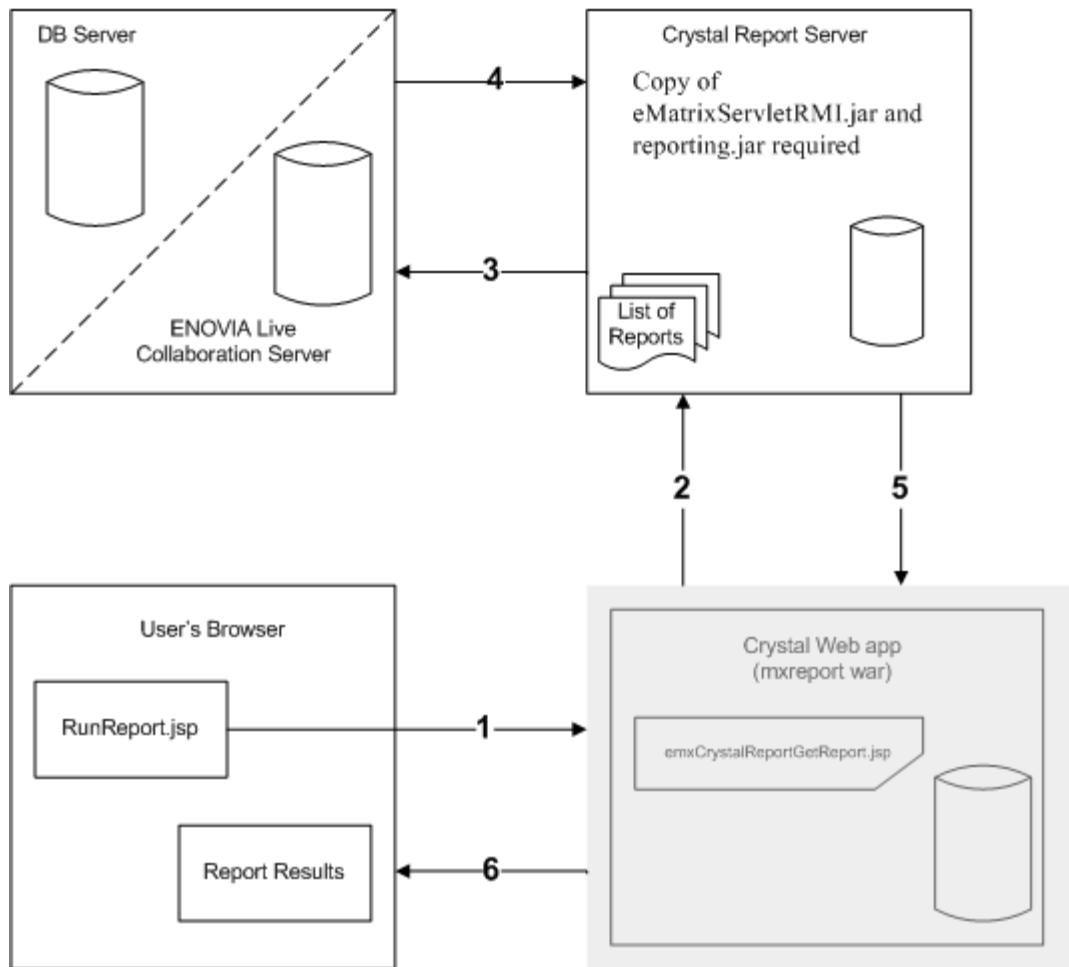
Core Support

ENOVIA Live Collaboration software provides a set of classes packaged as beans in the reporting.jar file that provides the Java interface for Crystal to communicate with an ENOVIA Live Collaboration Server to access the server as a data source, one of the key pieces of a Crystal Report.

All Crystal Reports include information about the data source to use as well as formatting information that defines how the output is to be presented. This report definition, performed within Crystal, is stored as an RPT file which contains the information necessary to execute the report.

The reports must use the classes defined and packaged by the core in the com.matrixone.reporting package which is in the reporting.jar Java archive.

The following diagram indicates the general processing flow for executing a report and what "systems" are involved in that execution.



Step	Description of Event / Data Flow
1	The user selects the name of a Crystal Report stored on the Crystal Report server and presses the Done button. The URL to access the emxCrystalReportGetReport.jsp is determined from a property file setting in emxSystem.properties. The Crystal Web application (mxreport.war) must run on an ENOVIA-supported application server.
2	The emxCrystalReportGetReport.jsp that is included in mxreport.war is forwarded based on a URL configured in emxSystem.properties. This JSP then tells the Crystal Report server to run the report.
3	The Crystal Report server requests the report data via the data source class that is available in the eMatrixServletRMI.jar.
4	The ENOVIA Live Collaboration Server responds to the Crystal Report server with data that is formatted in row sets.
5	The Crystal Report server responds to the request in Step 2 with the results so the report can be built in DHTML.
6	Report output is presented to the user on the machine where the request originated in Step 1.

Installing BusinessObjects XI Release 2

Business Objects XI Release 2 is a Web-based report management and delivery system used to navigate, print and export reports. You need to install it to use Crystal Reports.

1. Follow instructions in the BusinessObjects installation documentation.
2. Locate the Crystal Reports configuration file, `CRConfig.xml`, in the `C:\Program Files\Business Objects\Common\3.5\java` directory and open it for editing.
3. Edit the Classpath variable to include the path to the `eMatrixServletRMI.jar` file.

```
<Classpath>...;C:\em1\rmi\10.7\java\lib\eMatrixServletRMI.jar  
</Classpath>
```

4. Set the JavaBeansClassPath variable to the `reporting.jar` file.

```
<JavaBeansClassPath>C:\em1Crystal\reporting.jar  
</JavaBeansClassPath>
```

5. Save the `CRConfig.xml` file.

Installing Crystal Reports Professional

Crystal Reports is a thick-client report designer that can be used for creating your own reports that use data from the ENOVIA Live Collaboration database. You need to install it following the instructions in this section.

Important: ENOVIA Live Collaboration requires the JavaBeans Connectivity for a data source. Crystal Reports does not normally have this data source available as an option. However, if it is installed on the same server as BusinessObjects XI Release 2, it will be available. If not, for use with ENOVIA products, Crystal Reports XI R2 Advanced (or Developer) edition must be installed.

1. Execute the install for Crystal Reports Professional according to the Crystal Reports instructions. Be sure to include Java Data under Data Access.
2. Copy the eMatrixServletRMI.jar and reporting.jar files (version 10.7 or higher) to the machine where Crystal Reports is installed.
3. Locate the Crystal Reports configuration file, CRConfig.xml, in the C:\Program Files\Business Objects\Common\3.5\java directory and open it for editing.
4. Edit the Classpath variable to include the path to the EMatrixServletRMI.jar file.

```
<Classpath>...;C:\em1\rmi\10.7\java\lib\eMatrixServletRMI.jar  
</Classpath>
```

5. Set the JavaBeansClassPath variable to the reporting.jar file.

```
<JavaBeansClassPath>C:\em1\Crystal\reporting.jar
```

```
</JavaBeansClassPath>
```

6. Save the CRConfig.xml file.

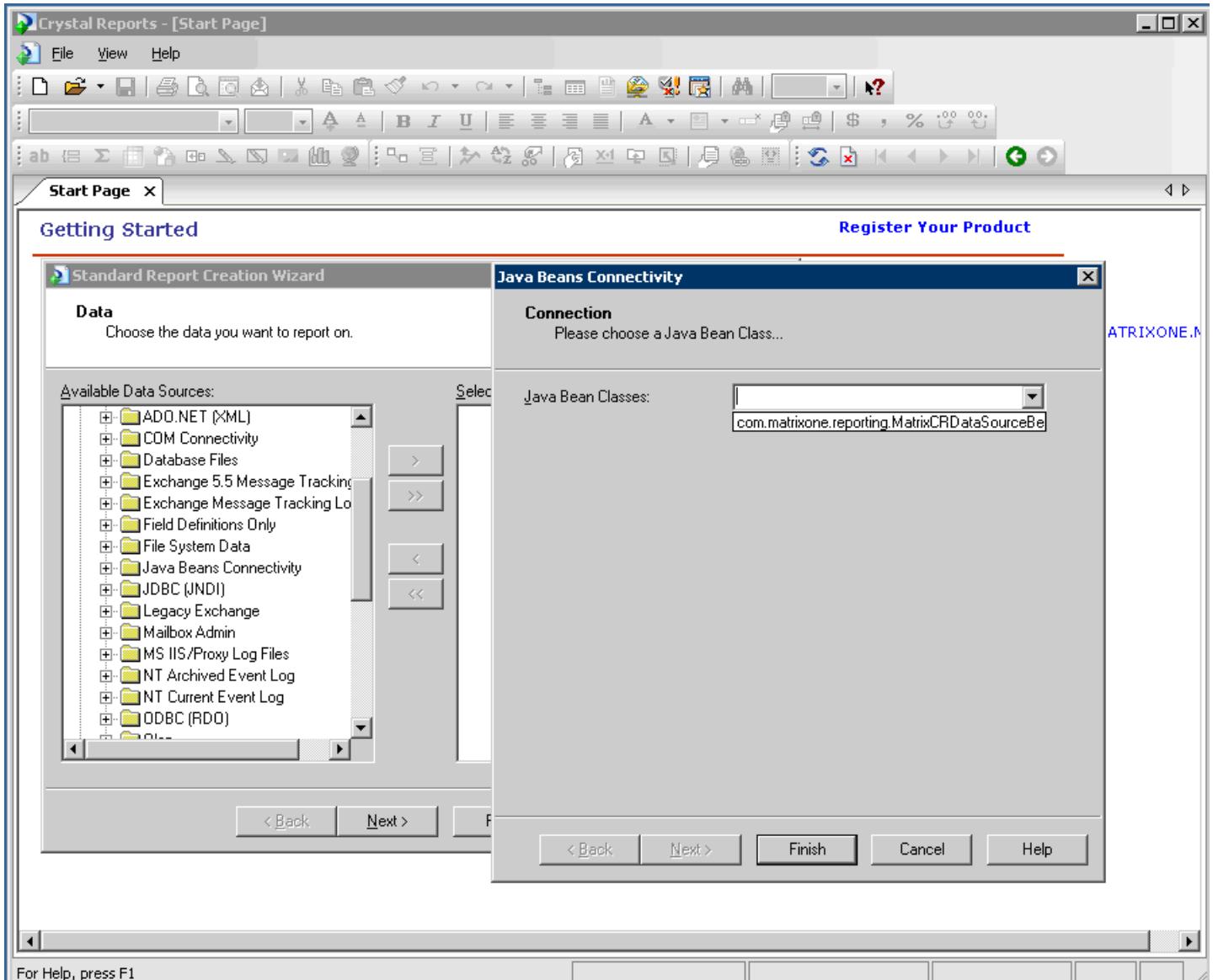
Verifying the Configuration

After installing the software and editing the configuration file, you should check that Crystal Reports can access the Java bean data source.

Before you begin:

You must install the Crystal Reports software. See [Installing Crystal Reports Professional](#).

1. Open the Crystal Reports client.
2. On the Start Page of the Report Creation Wizard, click the Java Beans Connectivity item.
3. In the dialog box, verify that the ENOVIA Live Collaboration classes are available in the pull-down list.



4. Click **Finish**.

Converting Crystal Reports 10

Reports created within Crystal Reports 10 need to be converted to work with Crystal Reports XI R2/Matrix 10.7. To do so, you need to update the datasource location, and update any scripts used with legacy reports.

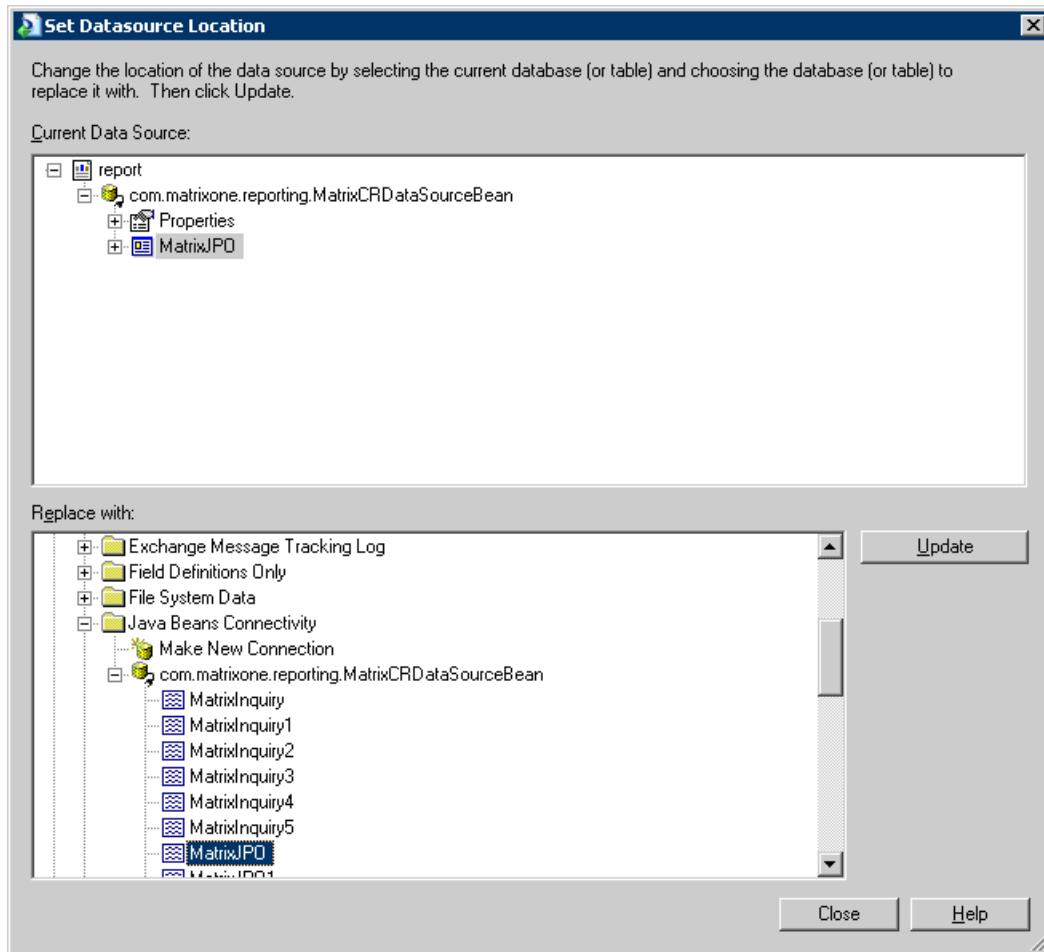
This task shows you how to:

- [Convert a Report](#)
- [Update Scripts Used with Reports](#)

Convert a Report

You can convert a Crystal Reports 10 report for use with ENOVIA products.

1. Open the report file using the Crystal Reports XI R2 report design.
2. Select Database > Set Datasource Location. The Set Datasource Location dialog box opens.



3. In the "Current Data Source" list, select MatrixJPO.

4. In the "Replace with" list, select MatrixJPO.

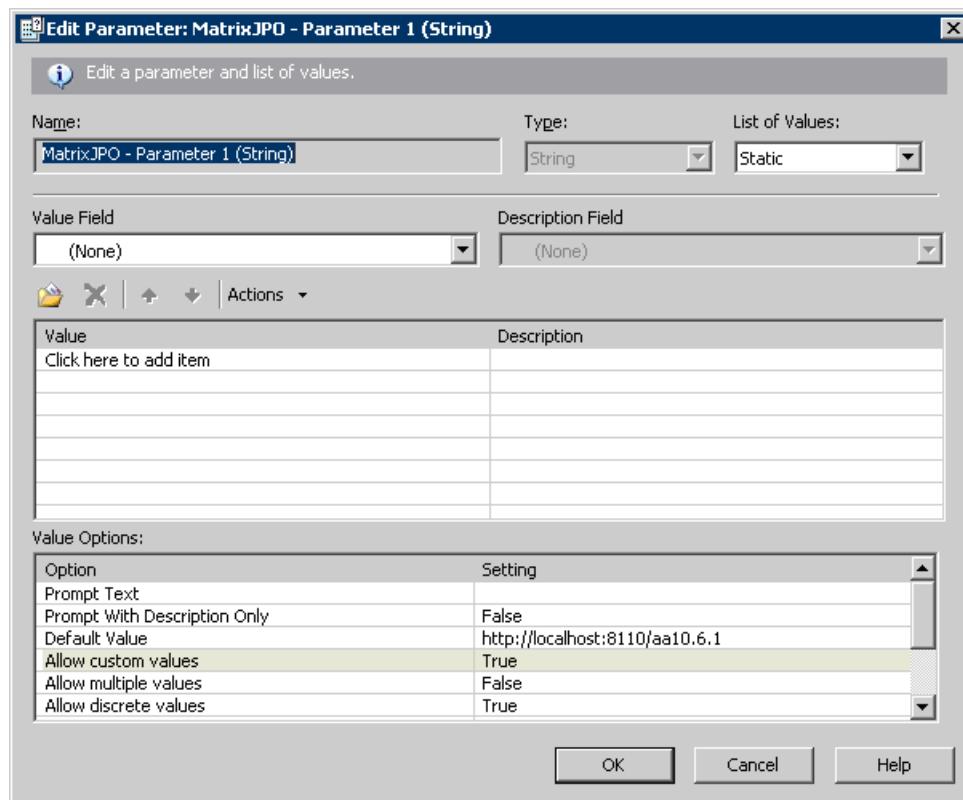
Although the names are the same, the "Current Data Source" list shows the MatrixJPO as configured for Crystal Reports 10, and the "Replace with" list shows the MatrixJPO configured for Crystal Reports XI.

5. Click **Update**.
6. Enter values for the java bean datasource parameters as required and click **Submit**.
7. Click **Close**.
8. If the report contains subreports, perform these steps for each subreport:
 - a. In the subreport section, select **Edit > Subreport**.
 - b. Repeat steps 2-7.
9. Save the report.

Update Scripts Used with Reports

Scripts for Crystal Reports XI include the datatype when referencing report fields; scripts for prior versions of Crystal Reports did not. For example, a Crystal Reports 10 field named "MatrixJPO - parameter 1" would need to be changed to "MatrixJPO - parameter 1 (String)" for the script to work.

1. To determine if you need to update scripts, open the report in Designer, and select Refresh Report Data. The software displays script errors indicating which scripts need to be updated (have errors).
2. To fix a field reference problem, edit the parameter indicated in the script errors message to add the datatype as a suffix to the field name.



3. Click OK.

Running Sample Reports

Business Process Services comes with three sample reports that can be used to test the installation.

For details, see [Sample Reports and JPOs](#). These sample reports (.rpt files) are included in ENOVIA_INSTALL/Apps/Framework/VERSION/common. Supporting inquiry(s) and JPO(s) are also installed with Business Process Services.

This task shows you how to:

- [Set Up Sample Report](#)
- [Run a Sample Report](#)

Before you begin: To run the sample reports from an ENOVIA product, BusinessObjects XI Release 2 needs to be installed. You can install Crystal Reports at a later time to design your own report files.

Set Up Sample Report

You can set up the sample reports for testing the installation.

1. Create a folder on the BusinessObjects server and copy the sample reports to it.
2. Publish the reports to BusinessObjects using the Crystal Management Console. For details, see the Crystal Enterprise XI Getting Started Guide.

Ensure that user "guest" has access to run the sample reports. By default, the emxCrystalReportGetReport.jsp uses the user name "guest" with no password and Enterprise authentication to connect to the Business Objects server. You can use different Crystal access if you change the user name and password in this file.

3. Modify emxSystem.properties to define the name of the Business Objects server, JSP file name to run report, list of reports (use sample report names). See the *Live Collaboration Administrator's Guide*.
4. Add the command AEFCrystalReportRunToolbar to the Toolbar menu. See [Menus and Toolbars](#) for details.
5. Deploy mxreport.war to the ENOVIA-supported application server that will act as the Crystal/ENOVIA gateway. Refer to your application server documentation for instructions on deploying a war file.



Run a Sample Report

Users assigned to the Administration Manager role can run Crystal Reports from within the ENOVIA products.

1. From any ENOVIA product, click **Tools > Run Report**.
2. Select the name of a pre-defined report.

The list of items in the combo box is obtained from the emxFramework.CrystalReport.ReportList property setting in emxSystem.properties.

3. Click **Done**.

The report is shown in a new browser window.

Modifying the System Properties for Reports

You need to modify the Crystal Reports section of emxSystem.properties on your ENOVIA Live Collaboration Server to define values necessary for the ENOVIA-Crystal integration.

1. Open the emxSystem.properties file for editing.

2. Locate these lines:

```
# Crystal Reports  
#Example: emxFramework.CrystalReport.GetReportPage = http://servername/appDir/  
emxCrystalReportGetReport.jsp  
emxFramework.CrystalReport.GetReportPage =
```

3. Enter the JSP page name that is used to run the report from the JSP environment.

By default, this property is assigned to empty string.

4. Locate these lines:

```
# ReportList - comma separated list of crystal reports  
emxFramework.CrystalReport.ReportList =
```

5. Enter a comma-separated list of report object names, available in the Crystal server, which can be run from an ENOVIA product environment.

By default, this property is assigned to empty string.

6. Locate these lines:

```
# Example: emxFramework.CrystalReport.ReportAppServer = servername  
emxFramework.CrystalReport.ReportAppServer =
```

7. Enter the server name where the BusinessObjects Server is running. The ReportAppServer is the CMS name.

By default, this property is assigned to empty string.

8. Enter the JSP page name that is used to run the report from the JSP environment.

For example:

```
# Crystal Reports  
#Example: emxFramework.CrystalReport.GetReportPage = http://servername/appDir/  
emxCrystalReportGetReport.jsp  
emxFramework.CrystalReport.GetReportPage = http://qesun2:7001/mxreport/  
emxCrystalReportGetReport.jsp  
# ReportList - comma separated list of crystal reports  
emxFramework.CrystalReport.ReportList = PartsByType,PersonCompany,ECRCategoryOfChange  
# Example: emxFramework.CrystalReport.ReportAppServer = servername  
emxFramework.CrystalReport.ReportAppServer = cystalserver1
```

Implementing the Crystal Reports Interface

The ENOVIA Crystal Reports Interface is delivered in the ENOVIACrystalReportsInterface.zip file located in the ENOVIA_INSTALL/Apps/ BusinessProcessServices/VERSION/Modules/ENOFramework/AppInstall/Programs folder.

1. Create a Crystal Report .rpt file based on the business use case, and define necessary parameters. You can configure the .rpt files to use either MatrixJPO or MatrixInquiry Java Bean data source. These Java Bean data sources provide the interface for the Crystal Report .rpt file and the ENOVIA Live Collaboration database through JPO or Inquiry.
2. If you configure the .rpt file to use MatrixJPO, you must create a JPO with a method name as specified in the .rpt file to get the data needed for Crystal Report. The JPO must return object of type "MatrixCachedRowSet." The full class name is com.matrixone.reporting.MatrixCachedRowSet. MatrixCachedRowSet extends javax.sql.RowSet and as such supports the full set of JDBC data types (INTEGER, REAL, DECIMAL, VARCHAR, CLOB, BLOB, etc.) and their corresponding updaters.
Or, if you configure the .rpt file to use MatrixInquiry, create an Inquiry to get the output needed for Crystal Report. The Inquiry object must conform to the guidelines explained in the following sections.
3. To test reports through Crystal Reports (rather than using the ENOVIA Run Report command in the Tools menu), configure the following in Crystal Reports Designer or through the BusinessObjects interface:

- URL (parameter 1) - XML Studio Customization Toolkit connection URL, for example, http://HOST:PORT/enovia
- Username (parameter 3) - ENOVIA user name
- Password (parameter 4) - ENOVIA password

The optional parameters (7-11 for JPO enabled data sources, 6-10 for Inquiry enabled) are passed to the JPO or Inquiry. How they are used is up to the JPO/Inquiry author. (ENOVIA products do not include pass any optional parameters for running reports. If needed you will have to customize it.)

Creating a Crystal Report File with the MatrixJPO Data Source

You can define a Crystal Report file to use the ENOVIA Live Collaboration Java Bean data source MatrixJPO delivered as part of the ENOVIA Live Collaboration distribution. This ENOVIA Live Collaboration Java Bean data source uses a JPO defined in the .rpt file as a parameter to generate the data.

Refer to the Crystal Reports documentation for details on creating .rpt files.

1. Create a Crystal Report file (*.rpt).
2. Enter parameter values based on this table.

Parameter Number	Parameter Value	Description
1	Url	Connection URL to ENOVIA (for example, http://HOST:PORT/enovia).
2	Cookies	Session authentication passed by the JSP page.
3	Username	ENOVIA user name, used for testing from Crystal Reports.
4	Password	ENOVIA password, used for testing from Crystal Reports.
5	JPO Name	JPO name, required to be set as a default in report.
6	JPO Method	JPO method, required to be set as a default in report.
7	arg1	Optional parameter defined in report .rpt file - will be available to the JPO.
8	arg2	Optional parameter defined in report .rpt file - will be available to the JPO.
9	arg3	Optional parameter defined in report .rpt file - will be available to the JPO.
10	arg4	Optional parameter defined in report .rpt file - will be available to the JPO.
11	arg5	Optional parameter defined in report .rpt file - will be available to the JPO.

The parameters "JPO name" and "JPO method" are mandatory and must be configured in the report file. These parameters can be set temporarily when testing through Crystal Reports, but must be reset and saved in order to run the report from the user interface. Other parameters are either passed in from the JSP or are optional.

Creating a JPO for the RPT File

After creating the .rpt file for a JPO, you need to develop a JPO to use the .rpt file and report data.

- Create a JPO with a method with the name specified in the Crystal Report file.

The JPO must return an object of type MatrixCachedRowSet. It can read all the input parameters sent from the report and make the query to the ENOVIA Live Collaboration database to populate the MatrixCachedRowSet object. The full class name is com.matrixone.reporting.MatrixCachedRowSet. MatrixCachedRowSet extends javax.sql.RowSet and as such supports the full set of JDBC data types (INTEGER, REAL, DECIMAL, VARCHAR, CLOB, BLOB) and their corresponding updaters.

Following is a simple example of a JPO method used to get the output:

```
public MatrixCachedRowSet runReport(Context context, String
[]args)
throws Exception
{
    static final SimpleDateFormat DEFAULT_DATE_FORMAT = new
SimpleDateFormat("m/d/yyyy h:mm:ss a");
    MatrixCachedRowSet mxrs = new MatrixCachedRowSet();
    MatrixRowSetMetaData mxrsMD = new
MatrixRowSetMetaData();
    int ncols = 2;
    mxrsMD.setColumnCount(ncols);
    mxrsMD.setColumnName(1, "name");
    mxrsMD.setColumnType(1, java.sql.Types.VARCHAR);
    mxrsMD.setColumnName(2, "originated");
    mxrsMD.setColumnType(2, java.sql.Types.TIMESTAMP);
    mxrs.setMetaData(mxrsMD);
    StringList objectSelects = new StringList(1);
    objectSelects.addElement("NAME");
    objectSelects.addElement("ORIGINATED");
    MapList ml = DomainObject.findObjects(context,
        "*", // type
        "*", // name
        "*", // rev
        "*", // owner
        "*", // vault
        null, // where
        false, // expand types
        objectSelects);

    mxrs.moveToInsertRow();
    Iterator mapI = ml.iterator();
    while (mapI.hasNext()) {
        Map m = (Map)mapI.next();
        mxrs.updateString(1, (String)m.get("name"));
        java.util.Date dt =
DEFAULT_DATE_FORMAT.parse((String)m.get("originated"), new
ParsePosition(0));
        mxrs.updateTimestamp(2, new
java.sql.Timestamp(dt.getTime()));
        mxrs.insertRow();
    }
    mxrs.moveToCurrentRow();
    mxrs.beforeFirst();
    return mxrs;
}
```

Creating a Crystal Report File with the MatrixInquiry Data Source

You can define a Crystal Report file to use the ENOVIA Live Collaboration Java Bean data source "MatrixInquiry" delivered as part of the ENOVIA Live Collaboration distribution.

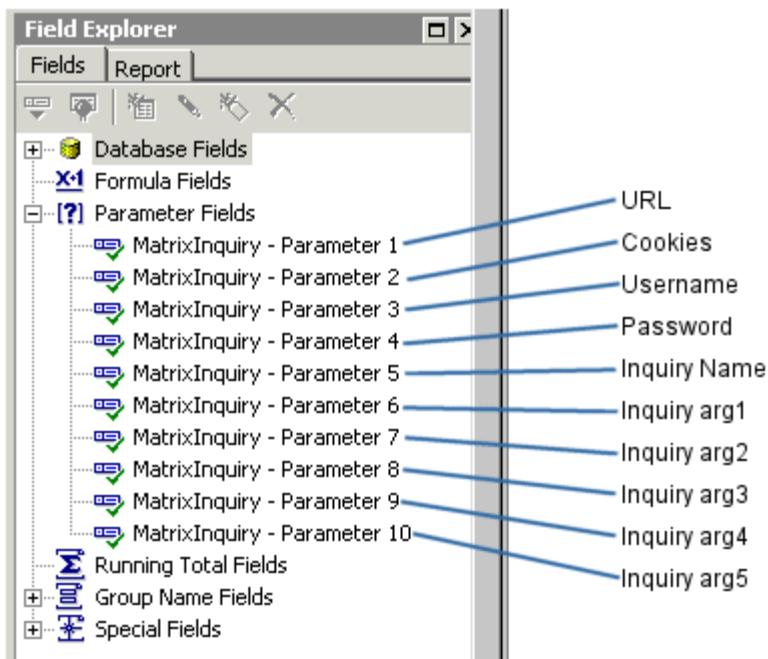
This bean is a simpler option than using a JPO because it requires no coding, but it is also less flexible. It does not allow binary data or multi-line strings. It uses a format string to describe the returned rows. Refer to the Crystal Reports documentation for details on creating .rpt files.

1. Create a Crystal Report file (*.rpt).
2. Enter parameter values based on this table.

Parameter Number	Parameter Value	Description
1	Url	Connection URL to ENOVIA (for example, http://HOST:PORT/enovia).
2	Cookies	Session authentication passed by the JSP page.
3	Username	ENOVIA user name, used for testing from Crystal Reports.
4	Password	ENOVIA password, used for testing from Crystal Reports.
5	Inquiry Name	Mandatory parameter to be defined in report .rpt file.
6	arg1	Optional parameter defined in report .rpt file - will be available to the Inquiry.
7	arg2	Optional parameter defined in report .rpt file - will be available to the Inquiry.
8	arg3	Optional parameter defined in report .rpt file - will be available to the Inquiry.
9	arg4	Optional parameter defined in report .rpt file - will be available to the Inquiry.
10	arg5	Optional parameter defined in report .rpt file - will be available to the Inquiry.

The parameter "inquiry name" is mandatory and must be configured in the report file. Other parameters are either passed in from the JSP or are optional.

This figure shows the Crystal Reports Field Explorer parameters and identifies the Inquiry name and other arguments.



Creating an Inquiry To Be Used by the RPT File

After creating the .rpt file for an Inquiry, you need to develop the Inquiry to use the .rpt file and report data.

- Create the Inquiry with the name specified in the Crystal Report file.

The MatrixInquiry Java Bean data source processes the Inquiry.

An inquiry used by a MatrixInquiry must conform to a very specific standard. The format consists of name-value pairs of data type keywords and variable names, separated by delimiters.

Each name-value pair is mapped to a data source column. Acceptable data type keywords are:

```
str - string  
strl - long string  
int - integer  
date - timestamp  
real - float  
bool - boolean
```

Variables are specified in the inquiry pattern. Here is a simple inquiry example:

```
inquiry SimpleInquiry  
    pattern '${TYPE}|${NAME}|${REV}|${OWNER}|${ORIGINATED}| \\\n        ${MODIFIED}|${DESCRIPTION}'  
    format 'str=${TYPE}|str=${NAME}|str=${REV}|str=${OWNER}| \\\n        date=${ORIGINATED}|date=${MODIFIED}|str=${DESCRIPTION}'  
    code 'temp query bus ECO,ECR * * select owner originated  
modified \  
        description dump |'
```

In Crystal Reports, using the MatrixInquiry data source with SimpleInquiry will show the following as a database table:

```
MatrixInquiry  
TYPE: String[254]  
NAME: String[254]  
REV: String[254]  
OWNER: String[254]  
ORIGINATED: DateTime  
MODIFIED: DateTime  
DESCRIPTION: String[254]
```

Sample Reports and JPOs

The following sample Crystal Reports (.rpt files) and the associated JPOs are distributed with the Business Process Services. They can be found in ENOVIA_INSTALL/Apps/Framework/VERSION/common.

The following topics are discussed:

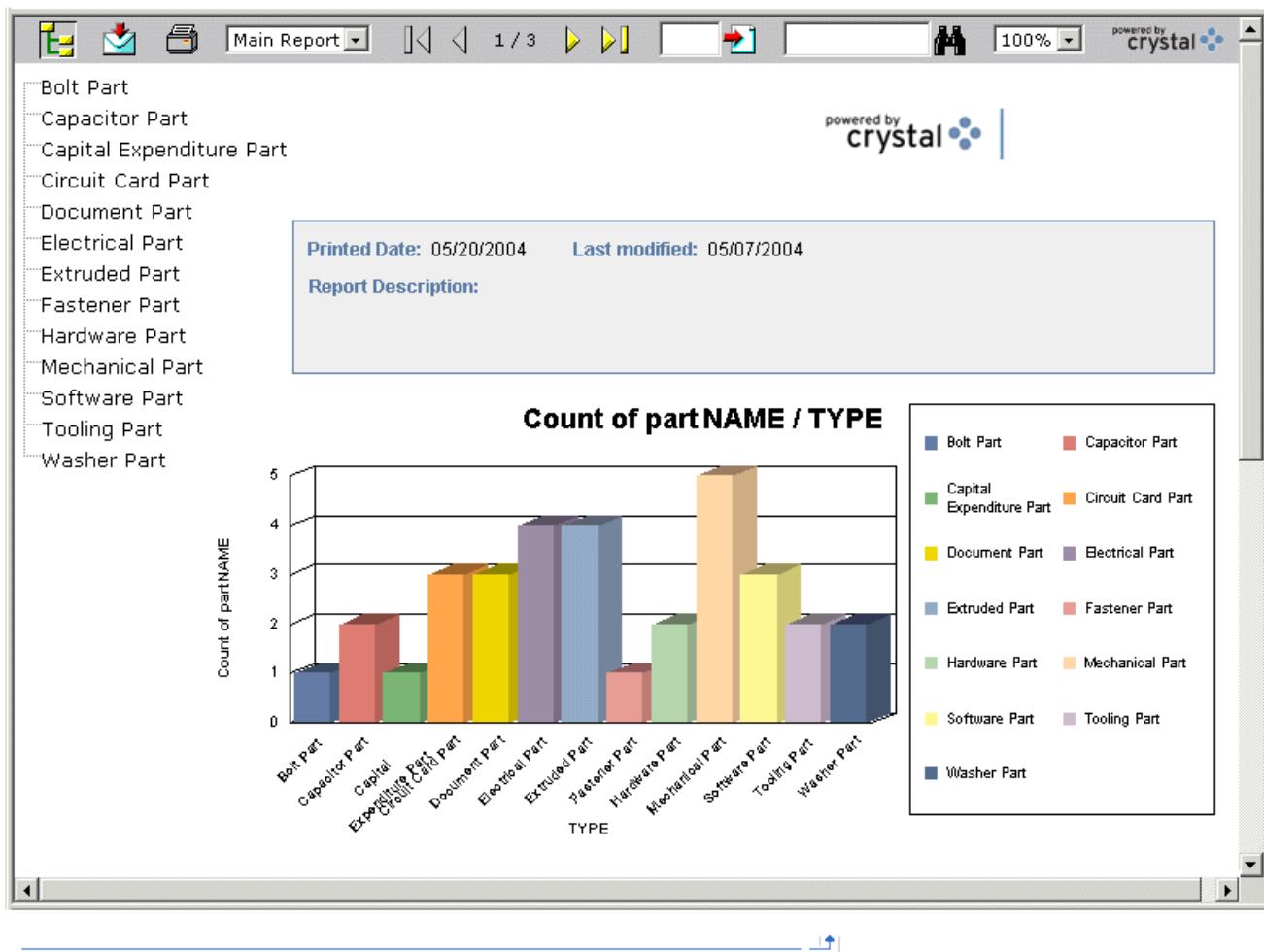
- [Sample #1 - Part Type Chart](#)
- [Sample #2 - Person and Company Pie Char](#)
- [Sample #3 - ECRs Category of Change Chart](#)

Sample #1 - Part Type Chart

This sample provides one JPO and a Crystal Report .rpt file to demonstrate a chart of part types. The JPO queries for all objects of type Part and its children types in the database.

The report file presents the data in a table format along with a bar chart that shows the number of different Part types available in the database.

The following screen shot shows the bar chart in the example report view.

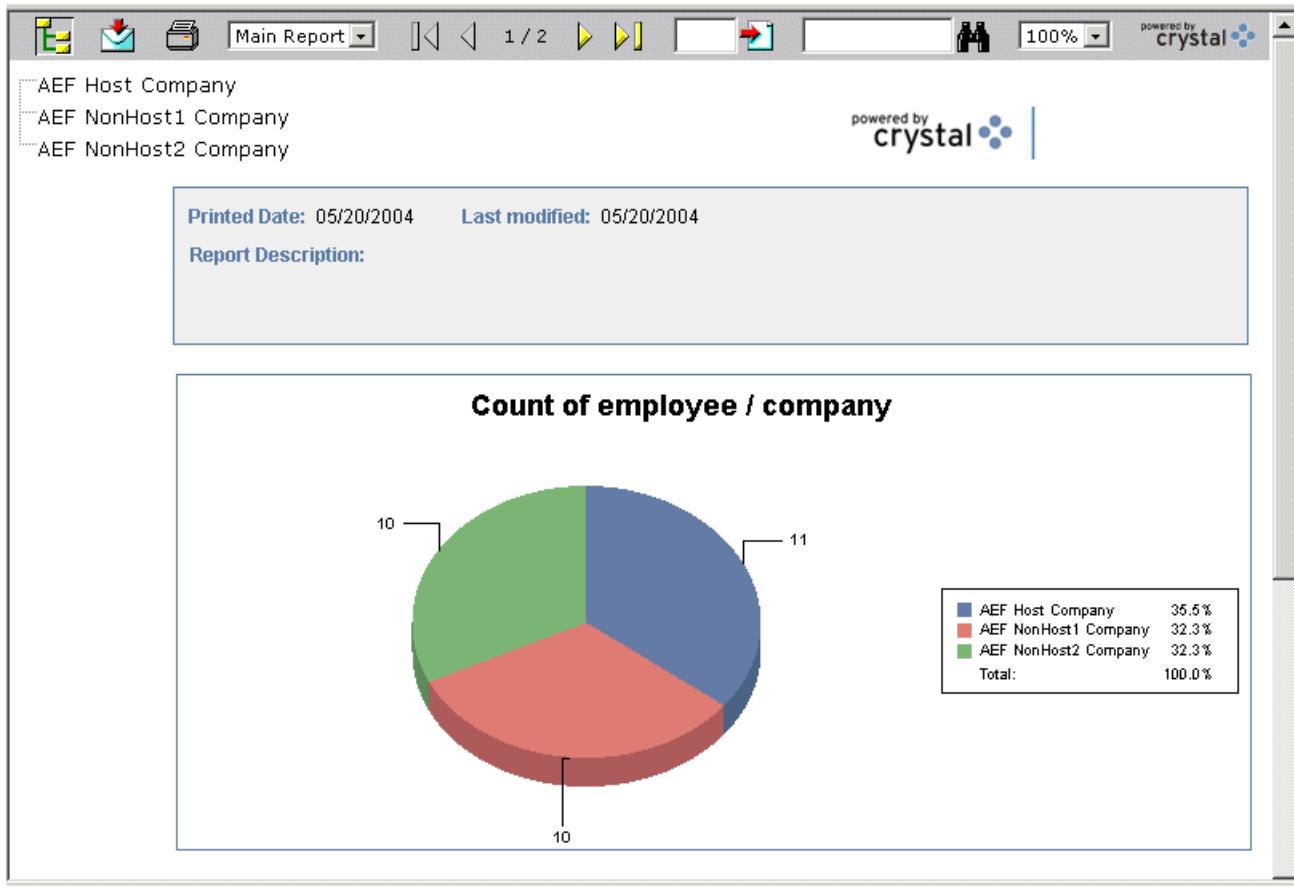


Sample #2 - Person and Company Pie Char

This sample provides a JPO and a Crystal Report .rpt file to demonstrate a pie chart of companies and employees. The JPO queries for all Person objects in the database and selects various attribute information. It then selects the Company objects that are connected via the Employee relationship.

The report .rpt file presents the data in a table format along with a pie chart that shows the number of employees per company.

The following screen shot shows the pie chart in the example report view.



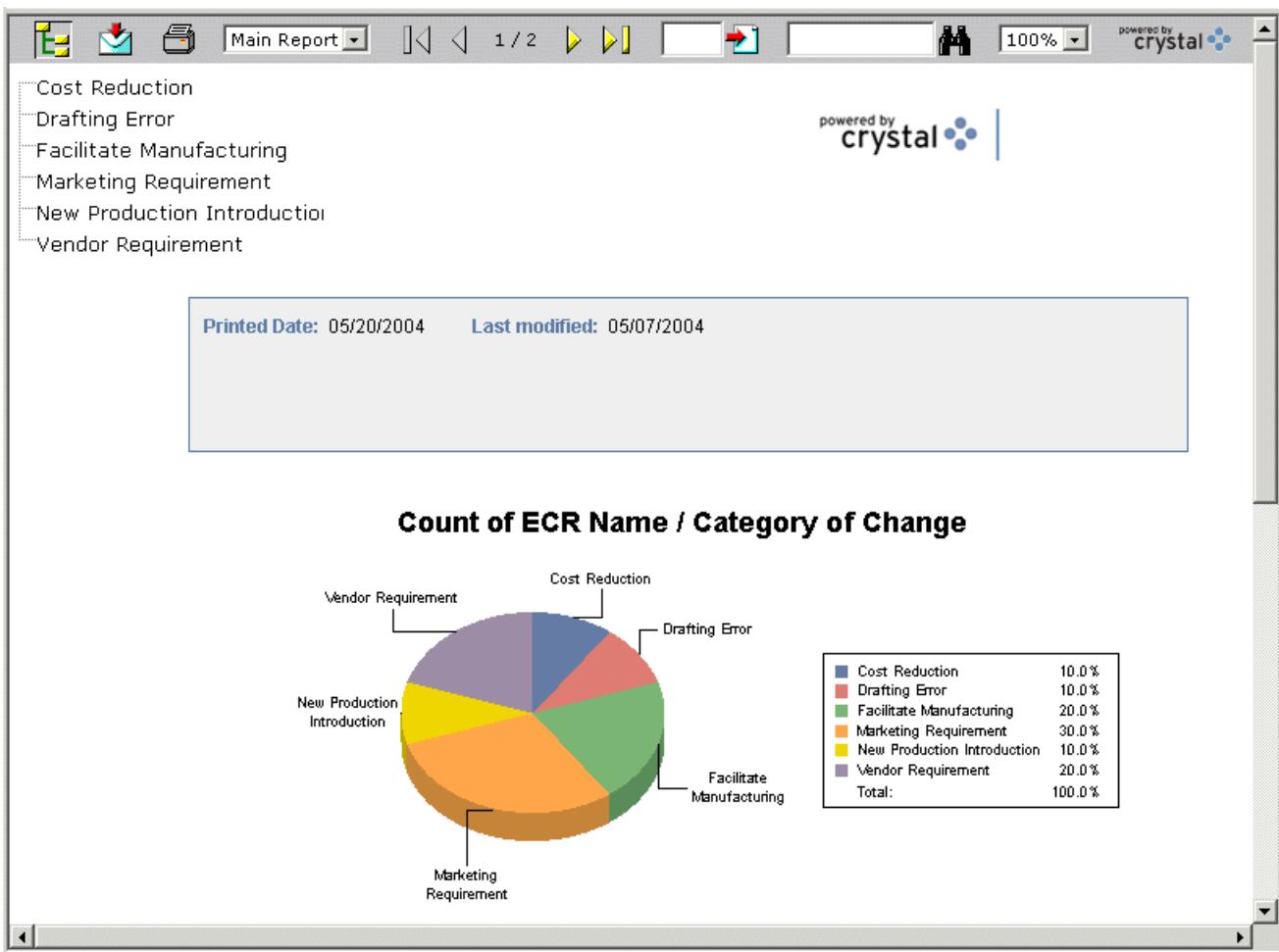
Sample #3 - ECRs Category of Change Chart

This sample provides one JPO and a Crystal Report .rpt file to demonstrate ECRs category of change.

The JPO queries for all ECR objects that have been created in the past 30 days and selects various attribute information. One of the selectable attributes is the "Category of Change" attribute, which contains an enumerated list of range values (Facilitate Manufacturing, Drafting Error, New Market Requirement, ...).

The report .rpt file presents some of the data in a table like view as well as creates a pie chart that shows the percentage of ECRs by Category of Change.

The following screen shot shows the pie chart in the example report view.



Menus and Toolbars

You can add and configure menus, such as the 3DS menu, Actions menu, global toolbar, and page toolbars. Menus present a list of commands that users can choose.

In this section:

- [Global Toolbar Menu](#)
- [Menus](#)
- [Right-click Menus](#)
- [Toolbars](#)

Global Toolbar Menu

The global toolbar shows at the top of all ENOVIA pages, and is the starting point for all applications. You should understand how menus are constructed before customizing them to meet your business needs.

Navigation trees that make up the Categories menu are considered another type of menu and are also represented by menu objects. They are described in [Navigation Trees](#).

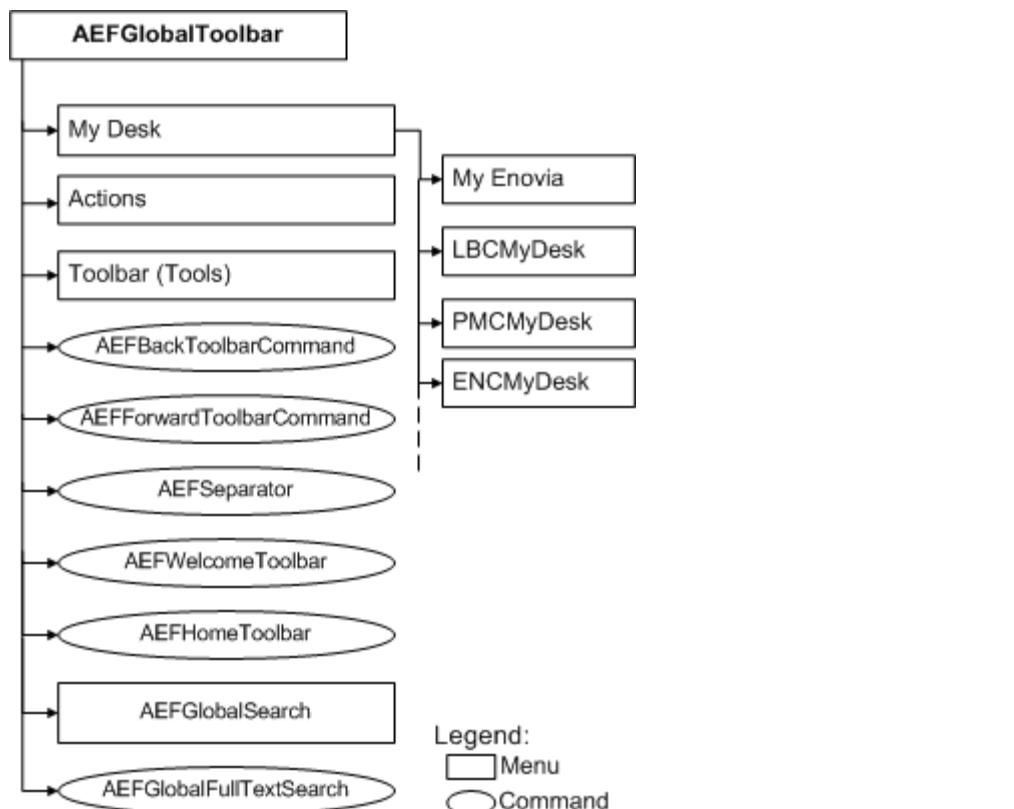
The following topics are discussed:

- [Global Toolbar Menu Construction](#)
- [3DS \(My Desk\) Menu](#)
- [My Enovia Menu](#)
- [Actions Menu](#)
- [Tools Menu](#)
- [Back and Forward Commands](#)

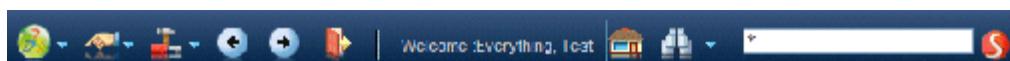
Global Toolbar Menu Construction

The global toolbar is the top-level toolbar to which the 3DS menu (previously named the My Desk menu), Actions, Back, Forward, and Tools menus, and other commands are connected.

The global toolbar menu is constructed as shown here.



The AEFGlobalToolbar menu looks like this in the ENOVIA products:



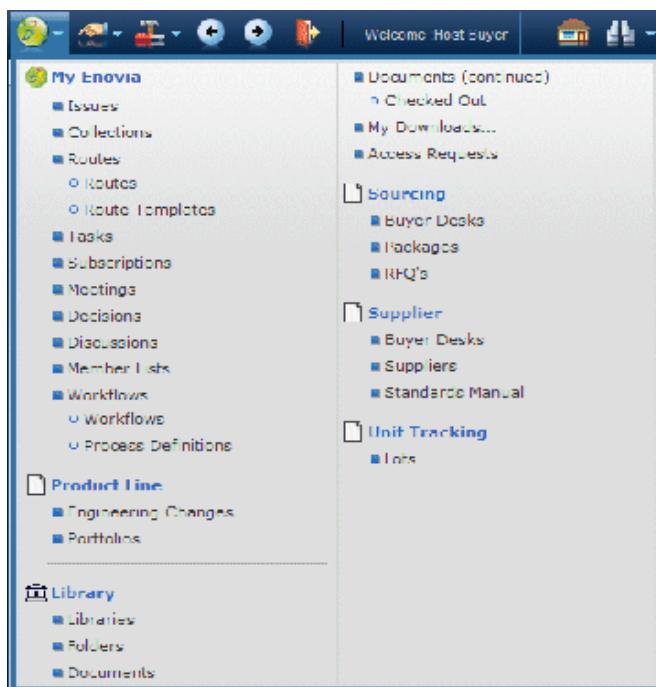
The AEFSeparator command simply displays a pipe symbol to visually organize the toolbar. Once a menu is expanded (by the user clicking the menu command from a toolbar), only commands can be clicked. The menus organize related sub-menus and commands.

3DS (My Desk) Menu

The My Desk menu contains commands for features shared by all applications, such as issues, collections and routes, plus each application has its own submenu within this menu. When the user clicks , the entire menu structure, expanded to all

sub-levels, opens. The user needs only one more click to select the needed command.

The 3DS menu shows as  on the toolbar.



The menus do not use a fixed width to display the contents. Instead, Business Process Services sizes the window based on the available space with a maximum number of 4 columns. If needed, a horizontal scroll bar is included to access additional columns.

The My Enovia menu includes commands and submenus that are part of Business Process Services and that can be used by all ENOVIA products.

The  (Dassault Systemes) button is defined by the value for the image setting:
Image=\${COMMON_DIR}/images/3dsButton.gif

You can remove the image (delete the value in Business Modeler) and replace it with a label. The Actions () and Tools () menus work the same way.

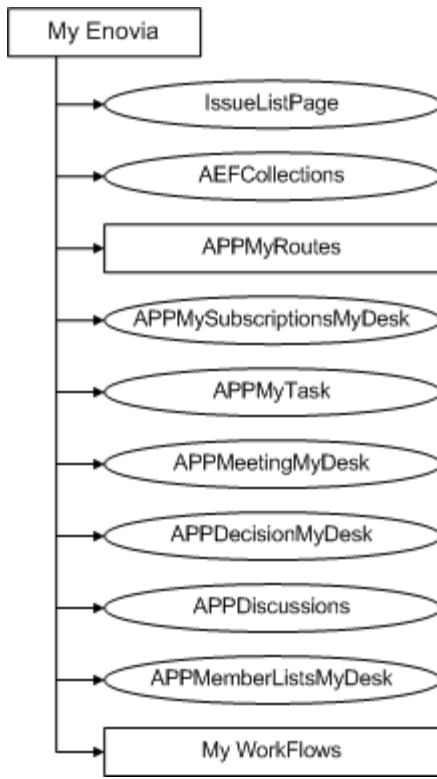
The "My Desk" metaphor dictates that the commands included in this menu should display existing information, as if users are looking through information on their desk. These menu items typically represent commands that present table pages. These table pages list objects that the user owns and/or needs to work on. When selected, these menu items display in the main frame and not in a new popup window.

My Enovia Menu

The My Enovia menu includes commands and submenus that are part of Business Process Services and that can be used by all ENOVIA products.

Business Process Services includes components that are used by multiple ENOVIA products, and can be used by custom applications. For example, any application can define Tasks, and then a user's task can be accessed from the My Enovia menu.

This diagram shows the structure of the My Enovia menu:



Ovals represent commands; rectangles represent submenus. APPMyRoutes contains a submenu for Route objects; My Workflows contains a submenu for Workflows objects.



Actions Menu

Commands in the Actions menu perform an action, such as creating a part, ECR, or product. Each application has a submenu with commands relevant to that application. When selected, Actions menu commands display in a popup window.

Actions menu commands perform actions that are not dependent on a selected item or the current context. Because users usually work within the context of a particular business item (a task, meeting, document, part, ECR, and so on), the commands for most actions appear on pages that pertain to a particular item or set of items and not on the global Actions menu. For example, one action a part owner can perform is to delete a part but the owner must first select the part to delete. Therefore, the command for deleting a part is on the Parts page.



Tools Menu

The Tools menu contains commands typically required by all applications and installed by Business Process Services.

For example, commands for changing the current user's password or preferences, managing profiles (People and Organizations), running reports (Metrics), and Utilities are in the Tools menu. The context user must have access to the tools for them to display in the menu.



Back and Forward Commands

The Back and Forward commands are disabled when a user logs in. As the user moves from page to page, the buttons are activated.

When the user clicks the Back or Forward commands, the ENOVIA system moves to the appropriate page. [Full-Text Search](#)

Menus

Business Process Services installs three menu administrative objects that represent the three types of menus in the dynamic UI: the My Desk menu, the Actions menu, and the Toolbar menu. You can use these to develop menus for your application.

In this section:

-  [About Building Menus](#)
-  [Building a Menu](#)
-  [Global Search Menu](#)

About Building Menus

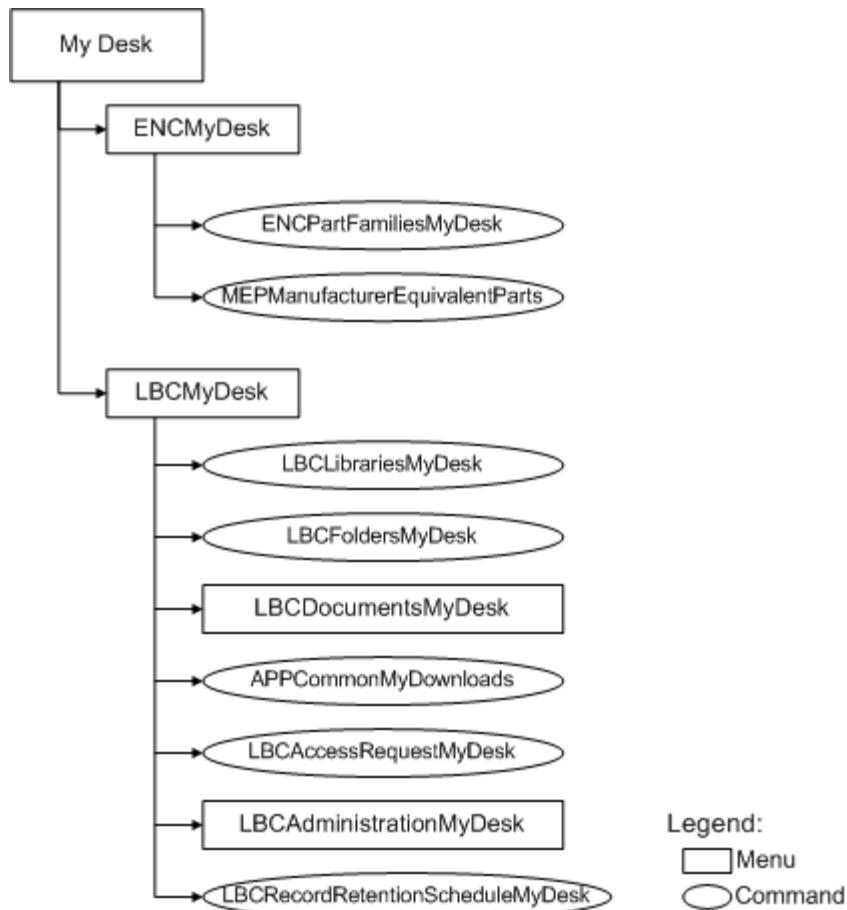
Business Process Services provides a mechanism for building menus that contain commands (and submenus with more commands, if needed) and attaching them to toolbars.

- [How Menus are Constructed](#)
- [Access to Menu Commands](#)

How Menus are Constructed

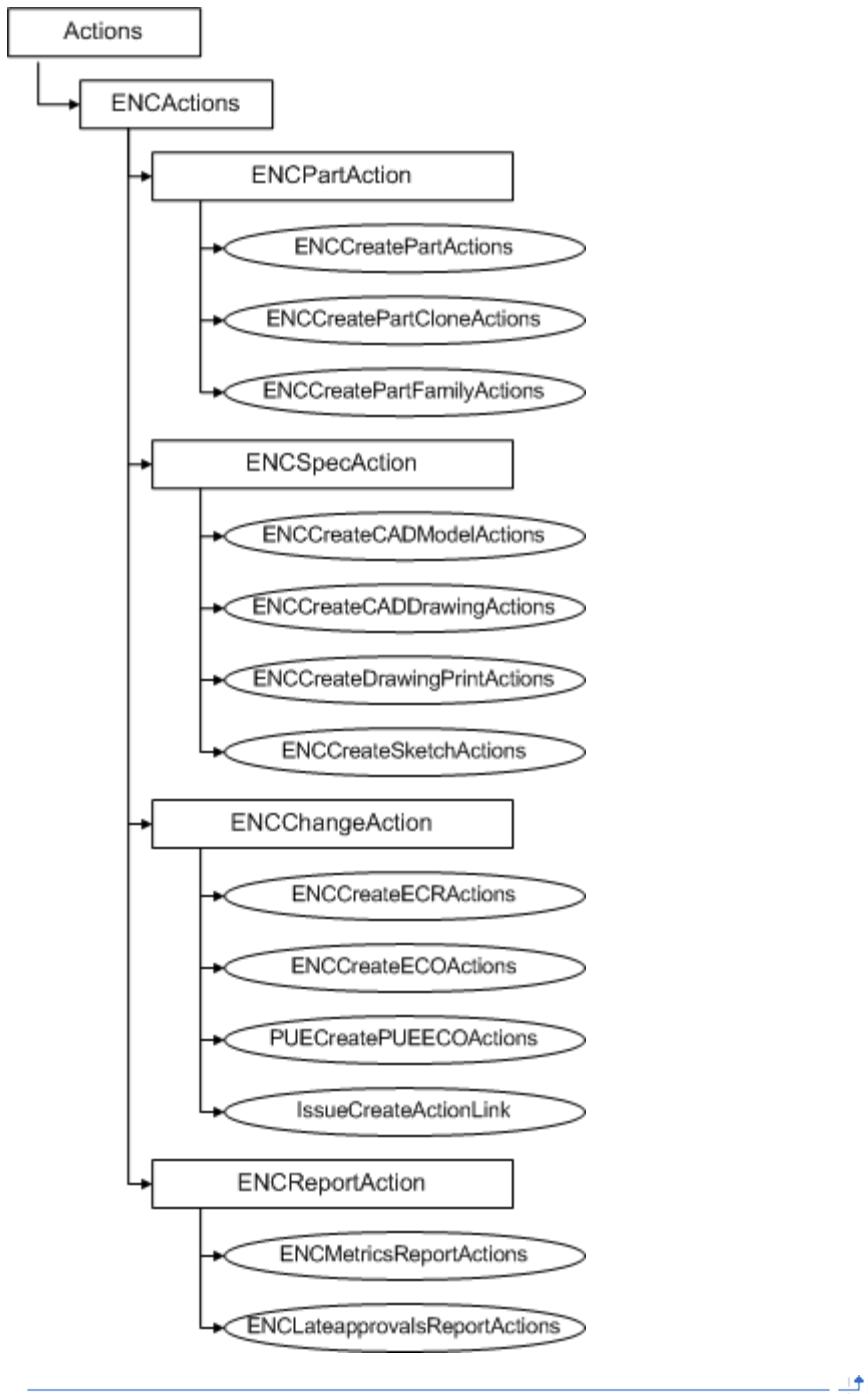
The My Desk menu object contains an additional menu object that represents the submenu for each installed application. Each of these submenu menu objects is in turn assigned a command or menu (to define another level of submenus) object for each link in the submenu.

The My Desk menu displays as  on the global toolbar. This graphic shows a sample administrative object structure needed for an installation that has Engineering Central and Library Central.



You can use Business Modeler to build the menus. Submenus and commands are created first, then added to a higher-level menu on the Items tab (in Business Modeler), or by using the equivalent MQL code. See the *Business Modeler Guide* for details on defining menus and commands.

Similarly, the definition for the Actions menu object should include a menu object that represents the submenu for each application. Each submenu should be assigned a command object for each link in the menu. In this example from Engineering Central, the ENCActions menu is connected to the Actions menu on the global toolbar. It has 4 menus connected to it as children (it could also have commands connected if appropriate), and then each of those 4 submenus has several commands connected to them.



Access to Menu Commands

You can define user access to menu commands.

[User Access to UI Components](#) describes how you can use the [Access Expression](#), [Access Mask](#), and [Access Program/Access Function](#) settings to determine whether the context user has access to an object. In this case, the object is a menu command.

The above settings return a true or false value. When true, the context user has access to the command. When false, you can use the [Access Behavior](#) setting to define whether the ENOVIA application hides the command (the command does not show in the menu), or disables the command (the command shows in gray in the menu and cannot be selected). The default setting is hide.

If a command that has child commands or submenus is disabled, the command shows in gray and cannot be expanded to access those children/submenus.

For some commands, such as Delete, if the user does not have delete access, you may want to hide the Delete command. For other commands, such as a Demote command that is not available for an object in its current state, you may want to disable the command. The user cannot access it when disabled, but knows that the functionality is there.

If a user does not have access to a command based on their Role instead of the above listed settings, then the [Access Behavior](#) setting is ignored and the command is always hidden.

Building a Menu

This section lists the main steps for building commands and menus for a custom application. Follow this procedure for creating menus for individual application pages, the global My Desk, Actions, and Search menus as needed to implement the custom application's functionality.

For details on parameters and settings available for menu and submenus, see [Parameters for Toolbar Menu Objects](#) and [Settings for Toolbar Menu Objects](#).

Before you begin:

Important: When customizing existing menus and commands and you do not want to include a defined menu/command, do not delete it. Instead, hide it using access controls. Future upgrades may be designed to update those menus/commands, and if they have been deleted, the update will fail.

The JSPs to execute the menu commands must have already been created.

1. Using Business Modeler, create a command object for every link that should be added to a menu:
 - a. Click **Object > New > Command**.
 - b. On the Basics tab, enter these details:
 - Name. You can use the underscore (_) and hyphen (-) characters, but no other special characters. For naming conventions, see [Naming Conventions for UI Administrative Objects](#).
 - Description
 - Label. The Label is the text that displays to the user when they open the menu.
 - c. On the Link tab, enter the name of the JSP that should be invoked when this command is selected.
 - d. On the Settings tab, enter name/value pairs as required for the command. At minimum, you need to define a value for the Registered Suite.
 - e. On the Access tab, define who can access this command. Typically, access is defined by role and not specific usernames. For example, anyone with a Librarian role has access to a command for creating a library.
 - f. Click **Create**.
2. Using Business Modeler, create a menu object to organize the commands created in Step 1. You can create as many menu layers as is appropriate for your application.
 - a. Select **Object > New > Menu**
 - b. On the Basics tab, enter these details:
 - Name. You can use the underscore (_) and hyphen (-) characters, but no other special characters. For naming conventions, see [Naming Conventions for UI Administrative Objects](#).
 - Description
 - Label. The Label is the text that displays to the user.
 - c. On the Link tab, enter the name of the JSP that should be invoked when this command is selected. Most menus will not have an Href defined; this option is sometimes used by application pages that define tree menus. See [About Navigation Trees](#).
 - d. On the Settings tab, enter name/value pairs as required for the command. See [Settings for Toolbar Menu Objects](#). At minimum, you need to define a value for the Registered Suite.
 - e. On the Items tab, add each command or menu required for this menu.
 - f. Click **Create**.
3. To see your changes in the user interface, log in as a user with the Administration Manager role and click the  > **Utilities > Reload Cache** and click the browser Refresh button.

The cache is refreshed automatically when the component age expires as defined in emxSystem.properties.

Global Search Menu

You can configure the global search menu to add specific searches, or to define which search is the default.

In this section:

- [About the Global Search Menu](#)
- [Parameters and Settings for AEFGlobalSearch Menu](#)
- [Parameters and Settings for a Command](#)

About the Global Search Menu

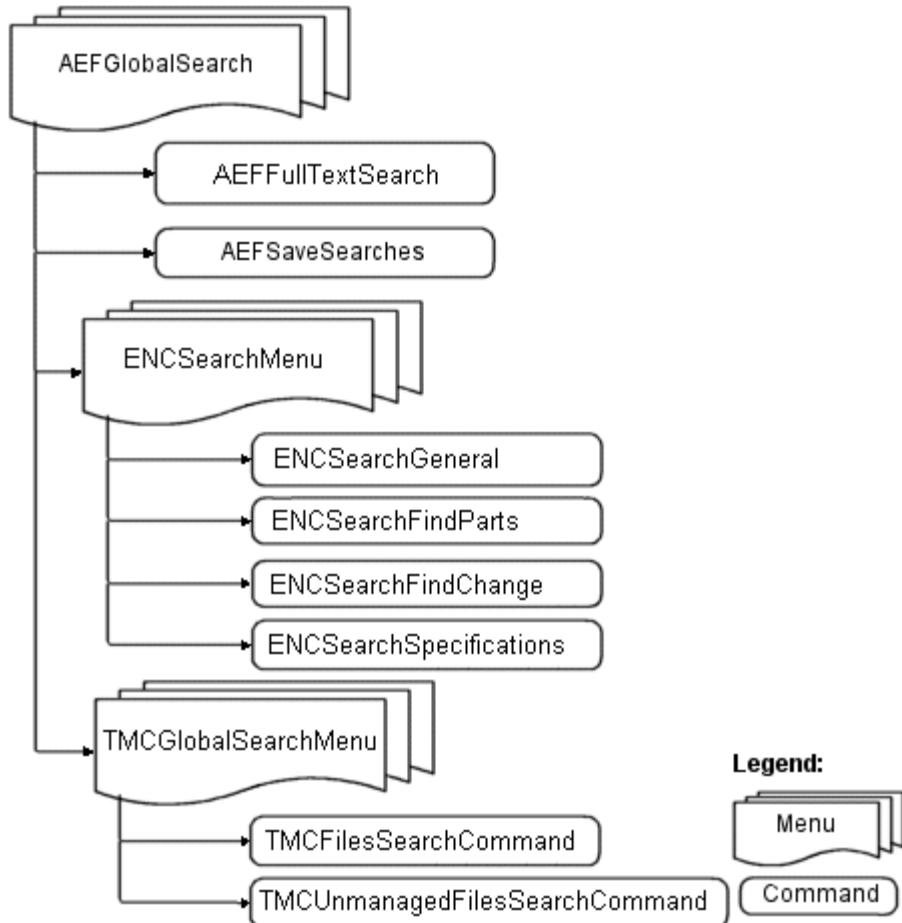
The top level Global Search menu, AEFGlobalSearch, holds all the Generic and Specific search menus/commands as shown in the following menu structure.

The following topics are discussed:

- [Structure of the Global Search Menu](#)
- [Commands for the Global Search Menu](#)

Structure of the Global Search Menu

This graphic depicts the global search menu.



Clicking this Global Search button opens the search window with the JSP page configured in the first command connected to the search menu AEFGlobalSearch. By default, the first command is the Full-Text Search defined using the command "AEFFullTextSearch."



Commands for the Global Search Menu

With only Business Process Services installed, the Global Search menu AEFGlobalSearch contains the commands, AEFFullTextSearch, AEFSavedSearches, and the menu for Team Central searches (TMCGlobalSearchMenu).

Applications create and connect the specific commands and menus to the appropriate location in the Global Search menu object as shown in the menu structure above.

For example, Engineering Central related searches are connected to one menu called ENCSearchMenu (only the first few commands are shown in the graphic) and this menu is connected to the framework Global Search menu AEFGlobalSearch. Similarly, all other applications group the searches and connect them as required. The commands and menus must conform to the requirements of the toolbar component.

Parameters and Settings for AEFGlobalSearch Menu

The following table describes the list of settings for menus in the AEFGlobalSearch menu.

Parameters and Settings	Description	Possible Values
Registered Suite	The suite that the menu belongs to.	EngineeringCentral
href	The URL to locate the search page.	emxSearch.jsp emxFullSearch.jsp
Image	An icon to display in front of the menu name.	\${COMMON_DIR}/images/iconSmallIPBlock.gif

Parameters and Settings for a Command

The following table lists the parameters and settings required to configure a command on the global search menu or a menu added to the global search menu.

Parameter and Settings	Description	Possible Values
Registered Suite	Suite Name for the search command.	EngineeringCentral TeamCentral DefectManagement
href	To be assigned to the search JSP page intended to display the search content. This specific JSP file must contain a JavaScript method called "doSearch()". The doSearch() method initiates the search process by calling a processing page. All original URL parameters passed-in when opening the search window are preserved and passed down to the specific search pages. The processing page must refresh the whole search window with the results page. The Results page may use the AEFSearchResultsToolbar as the toolbar to leverage "Revise Search" and "Add to Collection" support.	<code> \${SUITE_DIR}/emxDFTFormInclusionListSearch.jsp? type=Defect</code>
Target Location	The location where the search page displays.	windowshade searchContent
Help Marker	The helpmarker used to open context-sensitive help	emxhelpfullsearch
Popul Modal	Specifies whether the search page window is modal or non-modal	true false
Selectable in Preferences	Controls whether the link is available as a preferred Home page.	true false
Window Height	The height of the search page window in pixels.	630 540
Window Width	The width of the search page window in pixels.	850 1000

Right-click Menus

The right-click menu component defines the shortcut menu that pops up when a user clicks the right mouse button on a configured object type. Business Process Services provides a default right-click menu, and many ENOVIA products install type-specific right-click menus.

In this section:

-  [About Right-Click Menus](#)
-  [Dynamic Commands in a Right-Click Menu](#)

About Right-Click Menus

When defining an application, you can choose to use the Default right-click menu, or develop a custom menu.

See [About Building Menus](#) for details on defining whether a command should be hidden or disabled if a user does not have access to that command. The following topics are discussed:

- [Default Right-Click Menu](#)
- [Which Right-Click Menu Opens](#)

Default Right-Click Menu

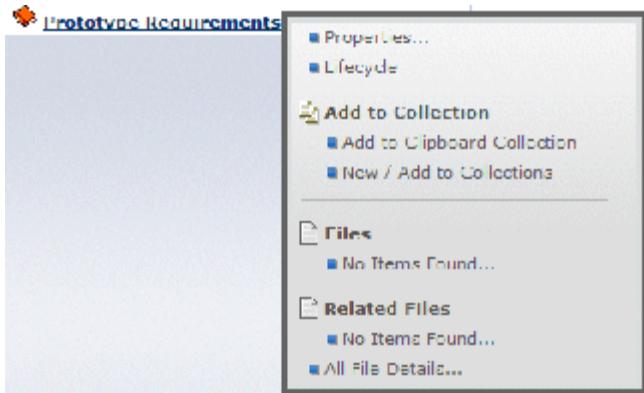
The right-click menu has a top-level component that holds one or more commands or additional menus. Any command that can be attached to a menu can be attached to a right-click menu. All settings and access controls for the toolbar component can be used with the right-click menu component.

These commands and menus are defined and attached to the top-level menu the same way as described in [Menus](#).

By default, all fields in the table or structure browser have the right-click component enabled. This feature can be disabled for the entire page by passing the `showRMB=false` URL parameter to the page.

To define a type-specific right-click menu, edit the type-specific menu, such as `type_Part`, and add the `RMB Menu` setting using the defined right-click menu name. You need to use the Business Modeler application to edit (add settings) to menu and column objects. If a column-specific right-click menu is defined, that menu overrides the type-specific menu. To define a column-specific right-click menu, use the `RMB Menu` setting on the column.

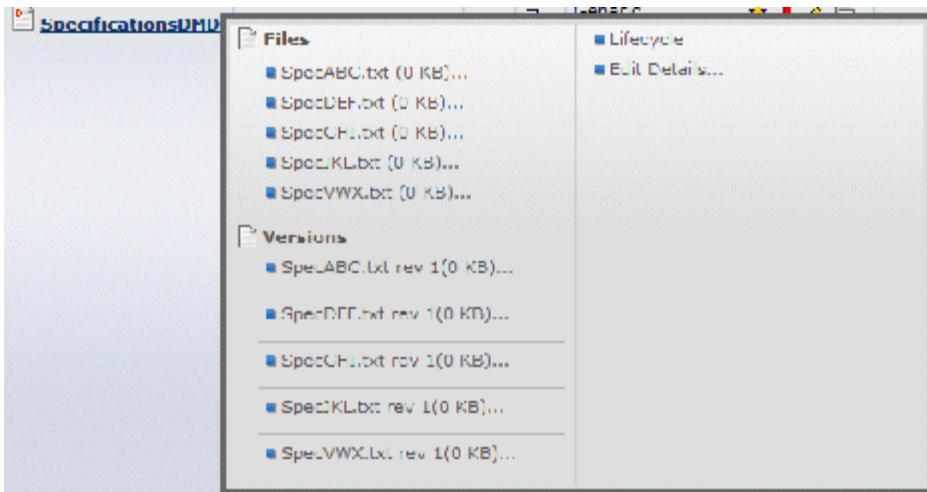
The default right-click menu, AEFDefaultRMB looks like this:



The AEFDefaultRMB menu includes these commands:

- **AEFProperties**--Opens the Properties page for the selected object.
- **AEFLifecycle**--Opens the Lifecycle page for the selected object.
- **AEFClipboardCollectionsRMB**--Submenu that includes these menu commands:
 - **AEFAddToClipboardCollectionRMB**--adds the object to the user's Clipboard collection
 - **AEFNewAddToCollectionsRMB**--opens the Select Collection dialog box where the user can select an existing collection or create a new collection
- **AEFSeparator**--Provides a visible separator for the menu.
- **APPQuickFileList**--Submenu that lists files checked into the selected object.
- **APPRelatedFiles**--Submenu that lists files checked into objects related to the selected object.
- **APPAllFileDetails**--Opens the Quick File list page.

The default Common Components right menu for DOCUMENTS (type_DOCUMENTS and all subtypes) is type_DOCUMENTSRMB and looks like this:



The type_DOCUMENTSRMB menu includes these commands:

- APPQuickFileList--Submenu that lists files checked into the selected object.
- APPVersionQuickFiles--Submenu that lists all versions of files checked into the selected object.
- AEFSeparator--Provides a visible separator for the menu.
- AEFLifecycle--Opens the Lifecycle page for the selected object.
- APPDocumentEditActionLink--Opens the Edit Properties page for the selected object.

You can add the AEFGenericDelete command to any of the default or custom shortcut menus.



Which Right-Click Menu Opens

Some columns may list different object types. Unless overridden by a column-specific right-click menu, the page opens the right-click menu for the type clicked on, and it may be a different menu for different items in the same column. Some table cells may contain multiple links. The component calls the right-click menu for the actual item right-clicked, and it may be a different menu for items in the same cell.

To determine which menu to popup when the user right-clicks on a field, the page uses these rules:

- If the field contains an object type, the type-specific right-menu is opened.
- If the column is configured with a right-click menu, that menu overrides any other right-click menu.
- If the field/column contains an attribute, the right-click menu for the row OID is used.
- If the field/column is defined with the `Alternate OID Expression` setting, the type-specific menu for the alternate OID is used.
- If the field/column is part of a group, (`Group Header` setting defined in the table), uses the OID for an object in the group (assumes the `Alternate OID Expression` setting). If the group does not include an alternate OID, the row OID is used.
- If the user clicks on a field that is not defined with a type-specific or column-specific right-click menu, the system looks up the type hierarchy until one is found, and if none are found, uses the system default right-click menu.

Dynamic Commands in a Right-Click Menu

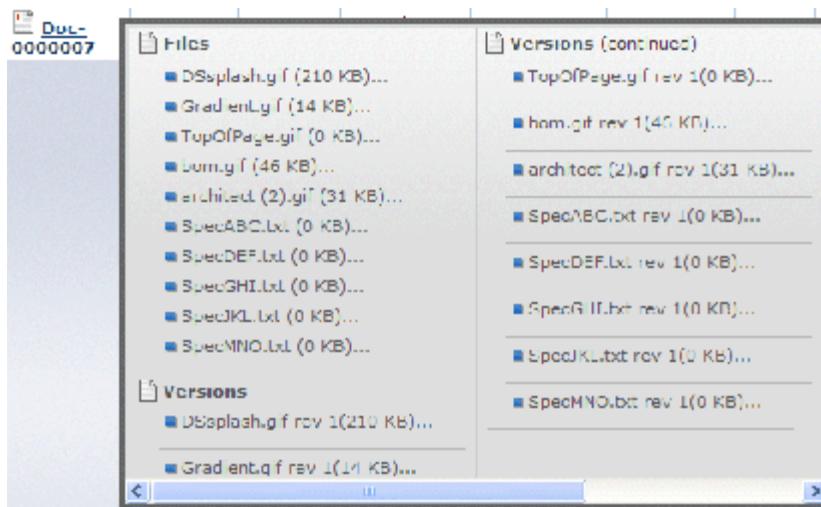
The right-click menu can include dynamic commands. Dynamic indicates that the content varies depending on the specific circumstances.

For example, the right-click menu for a document object could be configured to display the list of files checked into that object, and clicking on a file name opens the file.

Dynamic commands require a JPO:method to generate the command structure. The JPO:method is specified using the [Dynamic Command Program](#) and [Dynamic Command Function](#) settings on the command object. The menu is generated when the right mouse click event is fired. The command in the right-click menu, when clicked, call the page specified by the href of the command, which could be a popup or processing page depending on the Target Location specified for the command.

If these settings are configured on a menu, the returned items are listed as submenus/commands of that menu. If configured on a command, the returned items replace the current commands.

If the JPO:method returns a large number of items, a second column is added to the popup menu. The menu expands according to the current browser size; if needed, a scroll bar is added.



If your code contains complex logic or lengthy database calls, a [Loading...](#) message shows in the right-click menu until the data has been received and generated.

The Business Process Services (AEF and APP prefixes) install these dynamic menu commands that can also be used in customized menus:

- **AEFClipboardCollectionsRMB** menu--Opens a submenu that contains these commands:
 - **AEFAddToClipboardCollectionRMB**--adds the object to the user's Clipboard collection
 - **AEFNewAddToCollectionsRMB**--opens the Select Collection dialog box where the user can select an existing collection or create a new collection
- **AEFGenericDeleteRMB** command--Adds a Delete item to the right-click menu using the generic delete functionality.

This command uses `emxAEFUtil:getGenericDelete` as the JPO:method (Dynamic Command Program and Dynamic Command Function settings).

- **APPQuickFileList** command -- Opens a list of files checked into the object. The file names, used as the command labels, includes the file size in parentheses.

This command uses `emxAPPQuickFile:listQuickFiles` as the JPO:method (Dynamic Command Program and Dynamic Command Function settings) to generate the list of files.

- **APPRelatedFiles** command--Lists files checked into related objects. Related objects are determined based on the `emxFramework.QuickFileAccess.Relationships.Default` property in `emxSystem.properties`. The file names, used as the command labels, includes the file size in parentheses.

This command uses `emxAPPQuickFile:listRelatedFiles` as the JPO:method (Dynamic Command Program and Dynamic Command Function settings) to generate the list of files.

In addition, the `Pull Right=true` setting is specified.

- **APPReferenceDocumentQuickFiles** command--Lists files checked into related objects that are connected to the selected object with the Reference Document relationship. To add additional relationships for this command, edit the `Relationship Filter` setting on the command (default value is `from[<relationship_ReferenceDocument>].to.id`).

This command uses `emxAPPQuickFile:listReferenceDocuments` as the JPO:method (Dynamic Command Program and Dynamic Command Function settings) to generate the list of files.

In addition, the `Pull Right=true` setting is specified.

- **APPVersionQuickFiles**--Lists all versions of files checked into the object or checked into objects connected to it. Clicking a file name opens the user's File Download dialog allowing the user to save or open the file.

This command uses `emxAPPQuickFile:listFileVersion` as the JPO:method (Dynamic Command Program and Dynamic Command Function settings) to generate the list of files, with the `Access Expression=type.kindof=="$<type_DOCUMENTS>"` setting.

- **APPAllFileDetails**--Opens the Quick File window for the object, listing details for all files checked into this object or related objects.

This command uses `emxAPPQuickFile:getQuickFileTable` as the JPO:method (Dynamic Command Program and Dynamic Command Function settings) to open the Quick Files page for the object.

This code shows an example method signature:

```
public static List getRelatedFileItem(Context context, String[] args) throws FrameworkException
```

The JPO:method is provided a single args[] array containing a packed Map with these inputs:

- **paramMap**--Contains the request parameters passed to emxTable.jsp or emxIndentedTable.jsp stored as a map with request parameter names as keys. Also contains objectId, relId, etc.
- **commandMap**--Details of the dynamic command or menu.

The JPO:method returns a list with each entry corresponding to a menu command. Each entry in the list is a map with this structure:

Key	Description	Example Value
name	The name of the command object.	APPRelatedFiles
label	The text to display in the menu.	Related Files
href	The URL to execute when this command is clicked.	<code>#{COMMON_DIR}/emxTree.jsp</code> <code>#{SUITE_DIR}/emxEditPartDialog.jsp</code>
image	Icon to use as the toolbar command. By default, no icon is shown.	images/iconEditStatus.gif
alt	Mouse-over text for the command.	Related Files
settings	Key/value pairs of settings for the dynamic command. The key of the map is the setting name. See Settings for Toolbar Link Command Objects for a list of the supported settings. Typical settings used for dynamic commands include: Registered Suite Access Expression Access Program Access Function	
Children	A MapList that contains a HashMap for each command or submenu. Each map has the same structure as the parent HashMap.	

Toolbars

The configurable toolbar displays a drop-down menu of the actions that can be performed on a table page, form page, portal page, or a list of objects. You can customize existing toolbars, or you can create custom toolbars for your application.

In this section:

- [About Toolbars](#)
- [Input Controls](#)
- [Configurable Toolbar](#)
- [Building a Configurable Toolbar](#)
- [Toolbar Filters for the Structure Browser](#)
- [Parameters for Toolbar Menu Objects](#)
- [Settings for Toolbar Menu Objects](#)
- [Parameters for Toolbar Link Command Objects](#)
- [Settings for Toolbar Link Command Objects](#)
- [Implementing a Toolbar in a JSP](#)

About Toolbars

The configurable toolbar displays a drop-down menu of the actions that can be performed on a table page, form page, portal page, or a list of objects. The toolbar can also contain commands in the form of buttons, either specific to that page or default commands provided by Business Process Services.

The following topics are discussed:

- [Page Toolbar](#)
- [Drop-Down Menus](#)

Page Toolbar

This section describes the components of a page toolbar.

This graphic shows a toolbar that contains all the main components that a toolbar can have.



The toolbar can contain:

- Default links such as the Help and Printer-Friendly links shown above.
The toolbar component relies on querystring parameters to determine whether to display the Printer-Friendly link, the Export to Excel links, and the TipPage. By default, these buttons display, unless they are explicitly turned-off.
The Help link always displays. If no valid help marker is passed in, it defaults to the main help page for the specific application.
- Linked commands configured for the toolbar, such as the Create Issue link shown above.
When the user clicks such a link, the configured page for it is called.
- Top-level menu items that contain a list of drop-down actions, such as the Actions menu item shown above.
When the user clicks the menu item, a list of drop-down actions displays.
- Navigation tree including object categories, shown as the Categories menu. See [Navigation Trees](#) for details.
- Vertical dividers that separate links on the toolbar. The divider is specified by connecting the AEFSeparator command to the toolbar.

The above graphic has 2 vertical dividers. The dividers can be placed as needed. For example, you can configure the toolbar to have a divider for every two buttons. Or, a divider can be added to create a logical grouping of links, much like the grouping of new, open, save buttons on the Windows toolbar.

The emxFramework.Toolbar.PivotCommand.Limit property in the emxSystem.Properties file controls whether or not the Actions menu shows as a drop-down menu, or if the menu items display as toolbar buttons. If the number of menu items is less than or equal to the value of this property, the menu items will display as toolbar buttons. See the *Live Collaboration Administrator's Guide* for details.

This lists shows the URL Parameters that can be passed to the toolbar, emxToolbar.jsp, that enable/disable the specified toolbar buttons. For each parameter, the value can be true or false.

- AutoFilter
- customize
- CurrencyConverter
- editLink
- expandLevelFilter

If a toolbar menu includes the Expand Menu = true setting, the menu takes precedence over the expandLevelFilter URL parameter (the button will not be loaded in the toolbar and the custom Expand Menu will be loaded in the toolbar).

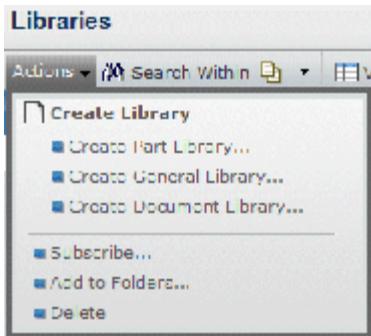
- export
- findMxLink
- massPromoteDemote
- multiColumnSort
- objectCompare
- PrinterFriendly
- showChartOptions
- showClipboard
- showTableCalcOptions
- tipPage
- triggerValidation

The helpMarker parameter is always passed. If no specific helpmarker value is provided, the button opens the help system at the cover page.

Drop-Down Menus

The drop-down contains the list of actions that can be performed from the current page.

This graphic shows a drop-down menu from the page toolbar:



Drop-down menus can contain:

- Top-level menu buttons that activate the drop-down menu. This is the Actions button in the graphic shown above. The title of the top-level menu button is configurable.
The key corresponding to the display name is found in the emxSystem.properties file by the name: emxFramework.UIActionBarMenu.Label. This property points to a key in the frameworkstringresource.properties file. The default value is emxNavigator.UIMenuBar.Shortcuts. To view or change the value for the display actions name, open the emxFrameworkStringResource.properties file and look for the emxNavigator.UIMenuBar.Shortcuts key and change its value to the desired string.
- Commands. When users click the links, the page configured for that link is called.
- Both the drop-down menu links and the toolbar links can be configured with the following characteristics:
 - Icons/Text/Both--For example, the Create Library link shown above is configured as an icon with text and the rest of the command links are shown as text only.
 - Activation--Toolbar buttons and drop-down menus can be enabled and disabled. For example, if the action requires that at least one item is selected on the list page, then the action is disabled until at least one item is selected.
 - Form Submission--Support for form submission through toolbar buttons/ drop-down options.
- Horizontal separators. Connect the command AEFSeparator to the menu.

The configurable toolbar replaces the top and bottom action bars used in prior versions of Business Process Services.

For both the toolbar and the drop-down menus, there are no set limits for the number of items.

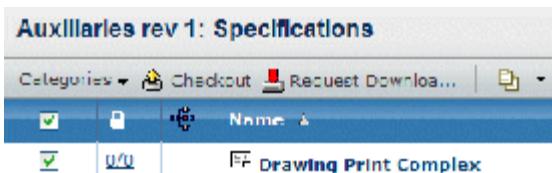
If you design a page that includes multiple toolbars, any drop-down menus must appear on the last toolbar passed to the page (becomes the bottom toolbar). Toolbars with many items that could force a drop-down menu into the overflow menu should be redesigned so that all drop-down menus appear on the bottom toolbar. If a drop-down menu appears on the top toolbar or is forced into the overflow menu, the drop-down menu will not work within the frameset.

Pivot Command

Drop-down menus can be converted to toolbar buttons.

When the Actions menu only contains a few submenu items, the items can display as toolbar buttons instead of as a drop-down menu. The emxFramework.Toolbar.PivotCommand.Limit property controls when the items in the Actions menu can be added to the toolbar. If the number of menu items is less than or equal to the limit, then the Framework places the commands in the toolbar instead of the menu (and the menu button does not display in the toolbar).

This screen shows a pivoted toolbar. The Specifications page does not have an Actions menu; the commands Checkout and Request Download, which normally would be in an Actions menu, have been added as buttons to the toolbar.



See the *Live Collaboration Administrator's Guide* for more information.

Overflow Menu

The ENOVIA toolbars can include a chevron symbol to support menu and commands that overflow the available space for the toolbar.

AA-EC-BOM-level-2a 1 : BOM PowerView

Categories 

Bill of Materials Where Used Spare Parts Markups

Actions  Edit All  BOM View      

engineering View Rev As Stored Deviation False Filter Add To >>

Name	Type	Rev	Policy	F/N	Ref Des	Compon	De
AA-EC-BOM-level-2a	Part	1	EC Part				

Input Controls

You can add 1 or more input control commands to the toolbar that take input and pass it to a JSP or JPO. The JSP could define a web form, table page, or other configurable component.

In this section:

- ❑ [Input Controls](#)
- ❑ [Textbox Input Control](#)
- ❑ [Textbox Input Control Examples](#)
- ❑ [Combobox Input Control](#)
- ❑ [Checkbox Input Control](#)
- ❑ [Submit Button Input Control](#)
- ❑ [Adding Input Controls to the Toolbar](#)

Input Controls

You can add as many controls as needed, however, ENOVIA recommends that you do not exceed 5 or usability and presentation could be adversely affected. All input controls must be first level commands attached to the toolbar.

The following topics are discussed:

- [Overflow Menu](#)
- [Input Values](#)

Overflow Menu

The overflow menu shows toolbar controls that do not fit on the toolbar.

When designing toolbars, any menus, buttons, or controls that do not fit are added to an overflow menu indicated by a chevron icon:



Important: Any input controls moved to the overflow menu will not function. Design the toolbar so that these controls do not get added to the overflow.



Input Values

To process the user's entry, you can use the href or Submit Program:Submit Function settings.

If you provide an href setting for the input control: When the user enters a value and clicks the command button, the system submits the page to the href URL, to the target location. The submit actions posts this data to the href URL:

- The names and values of all HTML input controls in the toolbar menus
- All request parameters that were passed to the main page
- For a flat table or structure browser, the timestamp to allow the processing JSP to retrieve data specific to the table (such as object IDs)

If the href calls a JSP, use `emxGetParameter(request,<commandName>)` to get the textbox value.

If the href calls javascript, use `document.getElementById(<commandName>).value` to get the textbox value.

If you provide an Submit Program:Submit Function setting for the input control: Write a JPO:method to process the values from one or more input controls.

Textbox Input Control

The textbox input control adds a box for user entry to the toolbar, and can be configured with a chooser and optional button.

The following topics are discussed:

- [Textbox Styles](#)
- [Textbox with Button](#)
- [Textbox with Chooser](#)
- [Textbox with Date Chooser](#)

Textbox Styles

When defining the `Input Type= textbox` control, you can include:

- textbox only
- textbox with chooser
- textbox with date chooser
- textbox with button

For toolbars with multiple input controls, when the user presses the Enter key, the following logic determines what happens:

- If the focus is in a textbox that has a submit button, the href for that textbox is fired.
- If the focus is in a textbox without a submit button, nothing happens. The user must explicitly click the submit button (assuming one is defined that takes the input from multiple input controls).



Textbox with Button

To include a textbox with a button, use these settings on the toolbar command.

- Input Type = textbox
- Action Label = button name
- href or Javascript

The `ButtonName` should indicate to the user the function of the input control. For example, if the input is used to search, the button should be labeled "Find" or "Search", depending on your convention. If the input is used as a filter, the button should be labeled "Filter". The submit button displays after the input control if the `Action Label` setting and `href` parameter (for the form or table object) are defined. When the input control has focus, the Enter key submits to the href parameter, even if the toolbar includes multiple input controls.

If you do not define an href or an Action label, no button is provided and nothing happens with the text a user enters in that textbox.



Textbox with Chooser

To configure a textbox with a chooser button, set `format=chooser`.

The href for the command should be a JSP that defines the chooser dialog. The page can be a Business Process Services-provided chooser page (such as the vault chooser), or a custom selection page. If the `Allow Manual Edit` setting on the command that defines the input control is set to `true`, the user can type data directly into the textbox (instead of clicking the ellipses button and opening the chooser page). When the user tabs out of the box or clicks elsewhere, the JSON code configured on the command is invoked.

For example:

```
 ${COMMON_DIR}/emxTypeChooser.jsp?SelectType=
multiselect&InclusionList=eServiceEngineeringCentral.Types
```



Textbox with Date Chooser

This setting adds the calendar tool after the textbox that allows the user to select a date. The user must use the calendar tool to enter a date.

To use a calendar control in a toolbar, the command must include the Target Location setting with a value of the content frame's name. For example: Target Location = `formViewDisplay`.

On the page that uses the textbox input control, you can also define a hidden parameter that assigns a name with the `_msvalue` suffix to the textbox. The system then assigns the value of the textbox to the hidden parameter (the value is calculated in milliseconds from midnight, January 1, 1970). The processing page can use this value for date comparisons or

validations.

Textbox Input Control Examples

These examples show the various ways you can configure a textbox input control for an existing or custom toolbar.

In all of these examples, the Label and Action Label parameters should be keys stored in string resource files to support internationalization. For convenience, the examples show the actual text that displays in the UI.

The following topics are discussed:

- [Example: Target Location is the Content Frame](#)
- [Example: Target Location is a Popup Window](#)
- [Example: Target Location is listHidden](#)

Example: Target Location is the Content Frame

This example shows how to configure a textbox input control that takes the user entry and opens a JSP in the content frame.

- Name: SearchTitle
- Href: emxSampleTitleSearch.jsp?mode=quickSearch
- Settings:
- Input Type=textbox
- Target Location=content
- Action Label=Find
- Registered Suite=Framework

See [Parameters for Toolbar Link Command Objects](#) and [Settings for Toolbar Link Command Objects](#) for details.

When a user enters text in the box and clicks the Find button, the content frame loads the page emxSampleTitleSearch.jsp to show the results of whatever processing is included in the JSP.



Example: Target Location is a Popup Window

This example shows how to configure a textbox input control that takes the user entry and opens a JSP in a popup window.

- Name: SearchTitle
- Href: emxSampleTitleSearchResultsDialog.jsp
- Label: "QuickSearch for:"
- Settings:
- Input Type=textbox
- Target Location=popup
- Action Label=Find
- Registered Suite=Framework

This example includes a Label, "QuickSearch for", that displays to the left of the text box to give the user a better idea of the function of the textbox. When the user enters text in the box and clicks Find, the JSP opens a new window and loads the results of the processing.



Example: Target Location is listHidden

This example shows a textbox command that uses listHidden as a Target Location.

- Name: filterTitle
- Href: emxSampleTitleSearchProcess.jsp
- Label: Value
- Settings:
- Input Type=textbox
- Target Location=listHidden
- Action Label=Filter
- Registered Suite=Framework

With the label "Value" and the button name "Filter", the user knows to enter a value on which to filter the table. When the user enters text and clicks the Filter button, the page posts the data to emxSampleTitleSearchProcess.jsp into the listHidden frame, where the page can alter the results and refresh the body page accordingly.

Combobox Input Control

You can configure a control for your toolbar with the Input Type defined as a combobox.

The defined `href` or `Submit Program:Submit Function` is executed for the `onChange` event of the combobox. The Action Label setting of the command shows as the Label.

When using the combobox input control, you can use one of these methods to populate the list:

- Range Program and Range Function settings
- Range Value setting
- Display Range Value setting

The Range Value option provides a comma-separated list of options that can be static text or string resource keys. The Range Program/Range Function settings define the JPO:method to use to build the list of choices for the combobox. These functions work the same as described in [Column Values Editable from Combo Box, Values from JPO](#).

If you provide both a Range Value setting and the Range Program/Range Functions settings, the Range Value setting is used. This example shows using a combobox as the input control.

- Name: ShowTaskFilter
- Href: emxShowFilteredTasks.jsp
- Label: Show
- Settings:
- Input Type=combobox

The `onChange` event triggers the `emxShowFilteredTasks.jsp`.

Checkbox Input Control

You can define a command on a toolbar to use a checkbox as the input control.

When using the checkbox input control, a single checkbox displays with values of true/false. Any Label defined in the component displays after the checkbox. When the user checks or clears the checkbox (the onClick event), the defined href or Submit Program:Submit Function is executed. You can use the `Default = true` setting for the command to have the checkbox selected by default.

This example shows using a checkbox as the input control.

- Name: ToRelationExpand
- Href: emxShowToRelatedObjects.jsp
- Label: "To"
- Settings:
- Input Type=checkbox
- Registered Suite=framework

The onClick event triggers the emxShowToRelatedObjects.jsp.

Submit Button Input Control

You can define a command on the toolbar to use a Submit Button as the input control.

The `Input Type=submit` allows you to define several input controls on the toolbar, and when the user clicks the Submit button, all of the values entered for any of the controls are submitted to the defined href/Javascript.

This example shows using a checkbox as the input control.

- Name: SubmitButton
- Href: emxShowDetails.jsp
- Label: "Submit"
- Settings:
 - Input Type=submit
 - Target Location=popup
 - Registered Suite=framework

When the user clicks the Submit button, the emxShowDetails.jsp is executed in a popup window.

Adding Input Controls to the Toolbar

You can use this procedure as a guideline for adding your own custom input controls to an existing or custom toolbar.

1. In Business Modeler, locate the command you want to add to the toolbar and open it for editing.
2. Click the **Settings** tab.
3. Enter a value for the `Input Type` setting:
 - textbox
 - combobox
 - checkbox
 - submit
4. Enter a value for the `Action Label` setting (such as a button name).
5. To define how the input values will be processed, enter a value for the `href` or `Submit Program:Submit Function` setting.
6. To provide a label for the input control, enter a value for the `Label` administrative parameter of the command object.

When the user enters a value and clicks the command button, the system submits the page to the href URL (to the target location) or JPO you specified to process the input values. Custom code must be written to read the input and perform any required processing. The code can display or alter the results.

Configurable Toolbar

Before you configure a toolbar, you need to know how it should be constructed and how to link it to an application.

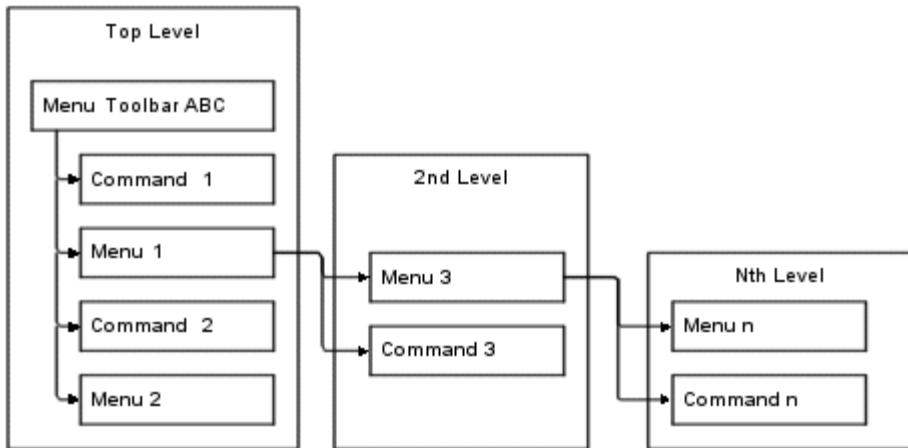
The following topics are discussed:

- [Construction of a Configurable Toolbar](#)
- [Toolbar Linked to an Application](#)

Construction of a Configurable Toolbar

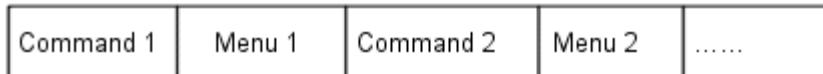
The configurable toolbar is configured using menus and commands. The toolbar supports an unlimited number of levels, but it is recommended that the number of levels be kept at 2 or 3.

The following image depicts one possible structure.



When the user clicks the top-level menu (Menu Toolbar ABC), the entire menu structure displays. Menu labels are not clickable, but the commands they include are. This setup allows the user to click once to open the menu structure, and then click one more time to select the needed option regardless of how many sub-menus are configured.

The top-Level section contains the toolbar menu and its connected menus and commands. The top-level menu, Menu Toolbar ABC in the above graphic, is an anchor for the toolbar structure. As such, it contains no settings. The connected menus and commands display as toolbar buttons on the top-level toolbar as shown below.



To display a standard link in the toolbar, one that does not contain a drop-down menu, you connect the command for the link to the top-level menu object. To display a drop-down menu on the toolbar, you connect the menu object for the drop-down menu, such as Menu 1, to the top-level menu object. Then connect the commands for the links within the drop-down menu to Menu 1.



Toolbar Linked to an Application

To use the configurable toolbar on a table, form, or portal page, the URL that calls the page needs to add the toolbar parameter.

`toolbar=TOOLBARMENU`

where `TOOLBARMENU` is the name of the toolbar menu. Below is an example URL that includes a toolbar on a table page:

`${COMMON_DIR}/emxTable.jsp?program=emxAEFCollection:getObjects&table=AEFCollectionItems&toolbar=AEFCollectionsToolBar&selection=multiple`

Building a Configurable Toolbar

You can design a custom toolbar and link it to an existing or custom application. This procedure lists the main steps needed for creating a toolbar and including it on a page.

1. Create a menu object to represent the top-level menu. See [Building a Menu](#).

Note: When naming commands and menus, you can use the underscore (_) and hyphen (-) characters, but no other special characters.

There are no settings needed for the top-level menu. For naming conventions, see [Naming Conventions for UI Administrative Objects](#).

2. Create a command object for each link that users should be able to access for the page, including those that should be within a drop-down menu.

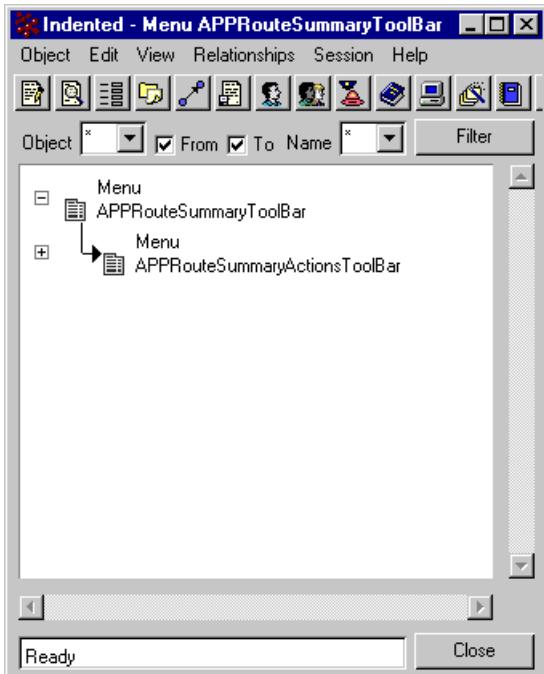
Business Process Services provides generic commands, such as `AEFGenericDelete`, that can be used in your toolbars.

For a description of how to fill in parameters and settings for toolbar link command objects, see [Parameters for Toolbar Link Command Objects](#). For naming conventions, see [Naming Conventions for UI Administrative Objects](#).

3. Create lower-level menu objects for each drop-down menu and assign the lower-level menu objects to the top-level menu object. For help understanding the structure, see [About Toolbars](#).

For a description of how to fill in parameters and settings for toolbar menu objects, see [Parameters for Tree Menu Objects](#).

For naming conventions, see [Naming Conventions for UI Administrative Objects](#).

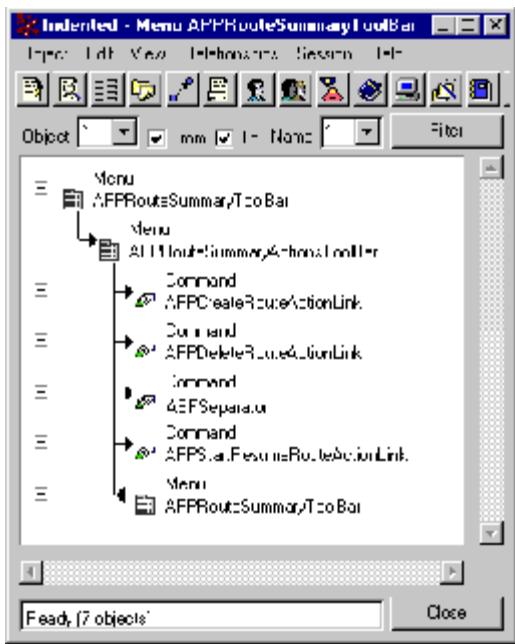


4. For each link that you want to appear on the toolbar, and not in a drop-down menu, assign the command to the top-level menu object that you created in Step 1.

Remember to assign commands to represent separators as needed. For each separator that you want in the toolbar, assign the command `AEFSeparator`, which is installed by default with Business Process Services, in the appropriate order for the top-level menu. When you add a separator command to the toolbar, it appears as a vertical line.

5. For each link that you want to appear in a drop-down menu, assign it to the appropriate menu.

Assign commands to represent separators as needed. For each separator that you want in the drop-down menu, assign the command `AEFSeparator` in the appropriate order for the menu. When you add a separator command to a drop-down menu, it appears as a horizontal line.



- In the URL that calls the table page, form page, or ENOVIA portal page that you want to include the toolbar on, include the parameter:

```
toolbar=TOOLBARMENU
```

Where TOOLBARMENU is the name of the toolbar menu. Below is an example URL that includes a toolbar on a table page:

```
 ${COMMON_DIR}/emxTable.jsp?program=
 emxAEFCollection:getObjects&table=AEFCollectionItems
 &toolbar=AEToolBarMenu&selection=multiple
```

Make sure you pass in the parameters as needed for the default links. The Help link always displays. If you do not pass in any parameters, the Printer Friendly and Export links display by default and the Tip and Conversion links do not. You can pass in parameters to have these links display or not display.

Printer Friendly=false/true

export=false/true

Tip Page=false/true

Currency Converter=false/true

- If you are working with the Web-based user interface as you are making changes and want to see your changes in the user

interface, click the  > Utilities > Reload Cache and click the browser's Refresh button.

The cache is refreshed automatically when the component age expires. This setting is in emxSystem.properties.

Only persons assigned to the Administration Manager role have access to the Reload Cache tool.

Toolbar Filters for the Structure Browser

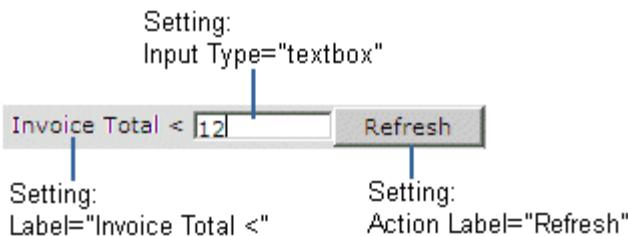
You can define a custom toolbar filter for a structure browser page. The custom toolbar is implemented by passing the toolbar URL parameter to emxIndentedTable.jsp.

See [Structure Browser](#) for details).

When defining a command for the custom filter toolbar, the command must be a first-level command. For usability issues, ENOVIA recommends that you do not use more than 5 commands on a toolbar. To define the size of the HTML controls (especially when using multiple controls), use the width setting (defined in pixels).

The toolbar must be defined as a menu that holds the commands that specify HTML controls (`Input Type of textbox`, `combobox`, or `submit`) to show in the toolbar. A date control (textbox with calendar icon) can be implemented using the `format=date` setting (no other format types are supported for custom filters). The controls are added to the toolbar in the order that the commands are listed in the filter menu. The toolbar can contain regular toolbar commands, such as the `AEFSeparator`. The command must have the `href` or `Code` setting specified that calls a JSP (for the `href` setting) or JavaScript (for the `Code` setting) that processes whatever the user entered in the control and updates the structure browser based on that selection.

When using a textbox or combobox, you can also define the `Action Label` setting. When defined, a submit button is provided following the textbox or combobox using the defined Action Label (which can be a string resource). Use the `Label` administrative parameter for the command to provide a text label in front of the input control. This figure shows a sample control with the settings required to draw it.



When defining the custom filter as a combobox, you can provide the values to populate the list using one of these settings:

- Range Function/Range Program
- Range Values
- Range Display Values

To provide a default value for the control, use the `Default` setting. The combobox control requires that the default value be one of the defined values.

The `href` or code called by the control is executed when:

- Textbox: pressing Enter
- Combobox: selecting a value
- Submit: clicking the button

The `href` defined for the command must be a JSP file in the web root directory, such as `${COMMON_DIR}/emxFILTER.jsp`. When a submit event occurs, the `href` is targeted to the frame specified in the Target Location URL parameter. The structure browser page submits all HTML controls specified in all toolbar menus. The JSP code needs to use the values or ignore as appropriate. The timestamp is also passed that defines where the JSP can retrieve the oids for the page.

This code shows a sample JSP for the `href` called by a filter command:

```
// Instantiate table bean (if required)
<jsp:useBean id="tableBean"
class="com.matrixone.apps.framework.ui.UTable"
scope="session"/>
<%
// Get time stamp id for current table from request
String timeStamp = emxGetParameter(request, "timeStamp");
// Get object & relationship ids
HashMap tableData = tableBean.getTableData(timeStamp);
MapList relBusObjList = tableBean.getObjectList(tableData);
// Get user inputs from HTML controls
String sTextboxValue = emxGetParameter(request, "<Name of
the Filter Command Representing text box>");
String sComboboxValue = emxGetParameter(request, "<Name of
the Filter Command Representing combo box>");
/*
**
```

```

    ** The code to filter out Object List and refreshing the
page will go here
    **
    */
%>

```

As an alternative to a JSP, you can write JavaScript code and enter it in the Code tab (in Business Modeler) for the command. This code must use JSON object syntax, such as shown in this example:

```

{
    main:function() {
        var elements=document.getElementsByTagName("input");
        for (var itr1 = 0; itr1 < elements.length; itr1++) {
            elements[itr1].removeAttribute('disabled');
        }

        elements=document.getElementsByTagName("select");
        for (var itr1 = 0; itr1 < elements.length; itr1++) {
            elements[itr1].removeAttribute('disabled');
        }
    }
}

```

If the only task you want the button to perform is to refresh the structure browser page, you can use this code:

```

{
    main:function() {
        filterPage();
    }
}

```

If both an href and JSON code is defined for a command, the JSON code takes precedence and the href link is ignored.

Parameters for Toolbar Menu Objects

This table details the available parameters for menu objects that represent the top-level menu for a toolbar and the lower-level menus.

For specific instructions on how to create menu objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Alt	Defines the text to display for a Tooltip.	--
Commands (specified in the Items tab in Business Modeler)	Defines the command objects that represent the links in the toolbar for the top-level menu and that represent links in the drop-down menu for lower-level menus. The order of the commands determines the order of the links on the toolbar and in the drop-down menu.	Names of command objects, such as: ENCCreateRevisionToolBarActionLink TMCEditProfileToolBarActionLink ENCEffectivityDateFilter
href	Defines the URL to call when a user clicks the label or image that represents the menu object. An href is not applicable to menu objects for My Desk, Actions, or Toolbar menus or submenus. The Toolbar menu cannot be clicked (only the individual tools can be clicked). The submenu labels for My Desk and Actions menus can be clicked but doing so only expands and contracts the menu and does not call another page.	--
Icon	Defines the icon for the object within ENOVIA Live Collaboration. The dynamic user interface does not display images for each application.	The name of an image file, such as Part.gif.
Label	Not applicable for top-level menus. Defines the text to display on the toolbar menu. This is the text that appears on the "button" that users click to access the drop-down menu, such as Action. Specify a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components . The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text.	Create New Part emxFramework.common.Actions If the label text is not found and no image is defined, the toolbar component displays an error icon to indicate that a configuration error has occurred.
Menus (specified in the Items tab in Business Modeler)	Defines the menu objects that represent drop-down menus. The order of the menus defines the order that they appear in the toolbar or drop-down menu. When defining custom filters, the list of menus to add to the toolbar. Menu names can use the underscore (_) or hyphen (-) characters, but no other special characters.	Names of menu objects, such as: ECBOMListToolBar TMCProjectsToolBar ENCFilterMenu
Settings	Defines additional settings for the behavior and appearance of the toolbar menu objects.	Name/value pairs, as defined in Parameters for Toolbar Menu Objects .

Settings for Toolbar Menu Objects

This table describes the available settings for menu objects that represent drop-down toolbar menus. There are no settings for the top-level menu. The name and value for each setting are case sensitive.

Setting	Description	Accepted Values/Examples
Access Behavior	<p>This setting works with the results of the Access Expression, Access Mask, or Access Program/Access Function settings. Those settings return a true or false value.</p> <p>When true, the menu command shows in the list and can be selected by the user. When false, this setting defines whether a menu command is hidden from the user (hide; does not display in the menu) or shown in gray (disable) so that it cannot be selected.</p> <p>This setting does not work with Role accesses. Commands not available to a user based on role are always hidden, regardless of the value of this setting.</p>	hide (default) disable
Access Expression Access Function Access Mask Access Program	Used to control access to tabs or to control other UI components. For details, see User Access to UI Components .	--
Dynamic Command Function	<p>Defines the method in the JPO specified by the Dynamic Command Program setting that returns a list containing the data structure to build the right-click menu. In general, the list contains dynamic options based on the user context.</p> <p>This setting is only used with the right-click menu component and the toolbar component.</p>	<JPO Method Name>
Dynamic Command Program	Defines the JPO invoked from a right-click menu component, or toolbar menu component.	<JPO Name>
Image	Specifies the image used for the menu. This is optional if you have defined a label. Required if there is no label.	\${COMMON_DIR}/iconSmallOrganization.gif
Maximum Length	Define the greatest number of characters that can be displayed on a drop-down menu label. The Title property of the button is used as the Alt text to display the full label.	Any number of characters, such as 25 When not set, the toolbar displays the entire label.
Mode	<p>Used for structure browser only.</p> <p>If defined for a toolbar command, that command is only enabled on the toolbar in the indicated mode (Edit or View). If not defined, the command is enabled in both Edit and View mode.</p> <p>If defined for a toolbar top-level menu, the entire toolbar only shows in the indicated mode and is hidden in the other mode.</p>	edit view
Pull Right	Determines whether the drop-down links display at the same level as the menu or pulled right to display the links in the next level. Applies only to second-level drop-down menus. For examples, see About Toolbars .	True (default) False
*Registered Suite	Specifies the application the toolbar menu belongs to. The system looks for files related to the menu in the registered directory for that application (specified in emxSystem.properties).	The value cannot contain any spaces. For example, EngineeringCentral or Framework. The value must be set to the suite name as defined in the key eServiceSuites.DisplayedSuites within

	<p>Based on the application name, the system passes the following parameters in the href URL:</p> <ul style="list-style-type: none"> suiteKey SuiteDirectory StringResourceFileId 	<p>emxSystem.properties. If the suite name starts with "eServiceSuite" then this prefix can be skipped and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite".</p> <p>In the href URL that is called when the menu is clicked, the system passes a parameter called "suiteKey". The value for the parameter is the property name from emxSystem.properties that maps to the setting's value.</p>
RMB Menu	<p>Specifies the name of the right-click menu to associate with the component.</p> <p>For a type-specific menu, such as type_Part, defines the type-specific right-click menu for that type.</p> <p>For a table column, specifies the right-click menu for that specific column.</p>	<admin menu name>

*Required

Parameters for Toolbar Link Command Objects

This table describes the available parameters for command objects used for toolbar links.

For specific instructions on how to create command objects using Business Modeler or MQLL, refer to the *Business Modeler Guide* or *MQLL Guide*.

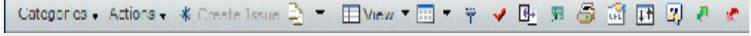
Parameter	Description	Accepted Values/Examples
Access	<p>Defines the persons, roles, and groups who can access the link or filter control. To make the link available to all users, regardless of role/group assignments, choose All.</p> <p>If no users are assigned access, the system assumes all users have access. If a user doesn't have access to any links in a menu, the menu is not displayed.</p> <p>Also see User Access to UI Components.</p>	Names of group, role, person administrative objects. or All (default)
Alt	<p>Defines the Alt text to appear over the link when the user mouses over it.</p> <p>When used with a custom toolbar filter, only use Alt if the Input Type = submit.</p>	--
Code	Custom JavaScript code entered in the Code tab of the command. JSON object syntax must be used for this code.	--
Description	A brief description of the function of the textbox command.	
href	<p>Defines the URL to call when a user clicks the link or submits input for a custom toolbar filter.</p> <p>When used to define the function associated with a textbox command, the href should be the JSP that processes the text entered by the user in the textbox.</p>	\${COMPONENT_DIR}/emxComponentsPage.jsp, \${SUITE_DIR}/emxTeamWorkspaceDetailsFS.jsp \${COMMON_DIR}/emxFiler.jsp
Icon	<p>Defines the Icon for the object within ENOVIA Live Collaboration.</p> <p>This parameter is NOT the image that displays on the dynamic user interface (for example, on a toolbar button). The Image setting is used to specify the UI image for components that have images associated with them. Many components, such as submenus, don't have images associated with them.</p>	The name of an image file, such as Part.gif.
Label	<p>Defines the text to display on the toolbar link. You can configure a link to have a text label, an image, or both. If you do not specify an image using the Image setting, you must specify a label.</p> <p>When used to define a textbox command on the toolbar, the Action Label and Input Type settings must also be defined.</p> <p>For a custom toolbar filter, defines the text that precedes the Input Type control.</p> <p>Either a string resource ID for the text string or the actual text string to display. To internationalize the text, use a string resource ID. See Internationalizing Dynamic UI Components. The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it does not find one, it displays the entered text.</p> <p>If the label text is not found and no image is defined, the toolbar component displays an error icon to indicate that a configuration error has occurred.</p>	Create New Part emxFramework.common.Actions emxFramework.common.Filter Apply Filter
Name	When using multiple textbox commands in the toolbar, the Name parameter distinguishes the specific textbox.	Find

	When used as a filter, the name of the filter toolbar item. The JPO and any custom JSP use this parameter to get the value entered by the user. The Name can include the underscore (_) or hyphen (-), but no other special characters.	ENCCustomFilter
Settings	Additional settings that define the behavior and appearance of the links. For a list of the accepted settings, see the table below.	Name/value pairs, as defined in Settings for Toolbar Link Command Objects .

Settings for Toolbar Link Command Objects

This table lists and describes the settings for command objects used for toolbars. The name and value for each setting are case sensitive.

Setting	Description	Accepted Values/Examples
Access Expression Access Function Access Mask Access Program	These settings control access to toolbars and other UI components. For details, see User Access to UI Components . Access Expression and Access Mask are not supported for non-object based structure browsers. Access Mask is not supported for toolbar custom filters.	--
Action Label	A text string shown as the button label after the textbox of an input control. Used only for Input Type = textbox or submit. This setting does not define a label to display in front of the input control. If this setting is not defined, then the button after the input control does not display. An href must also be provided when using this setting. For internationalization, a string resource key can be used. If a key is used, the Registered Suite setting must also be defined. The href parameter is required for the button to be displayed next to the text box.	emxFramework.SampleFeature.Find Find emxFramework.Common.Filter Apply Filter
Action Type	Tells the configurable toolbar that the command is a separator instead of a link. If the setting is not included, the toolbar treats the command as a standard link.	Separator--A line that separates one or more links. If the command is assigned to the top-level menu and is therefore on the toolbar itself, it is considered a vertical separator. Otherwise, it is considered a horizontal separator.
Allow Manual Edit	Only used when the format=chooser setting is defined, which is only used when the Input Control = textbox setting is defined. When set to false (the default), the user must click the ellipsis button to open the chooser dialog to select values for the textbox. When set to true, the user can enter text directly into the textbox, or use the chooser dialog.	false (default) true
Confirm Message	Provides a JavaScript confirmation message when users click on the link. For example, a custom delete	The actual text to display in the confirmation message or a string resource property key. To internationalize the message, a string resource key must be used:

	<p>message can be configured for the onClick event to display a JavaScript confirm message to the user. The Confirmation dialog has the "OK" and "Cancel" button. Clicking OK proceeds with the processing and Cancel cancels the operation.</p> <p>You can include the number of selected table rows in the message using a macro. See Using Macros and Expressions in Dynamic UI Components.</p>	<p>Are you sure you want to delete this object? emxFramework.common.alertMsg</p>
Default	<p>Provides the default value for the input control if the user does not enter a value but executes the control.</p> <p>The value can be static text or string resource keys.</p>	emxFramework.CustomControl.Default
Field Size	Width of the toolbar control in pixels. This setting helps design the toolbar if many controls need to be used in a small area.	100
format	Used only with Input Control = textbox when the entered text should be a date or a chooser. When date is specified used, the calendar tool shows after the text box. When chooser is specified, the browse button shows after the text box.	<p>date--Displays the column values as a date. Uses the tag lib "emxUtil:IzDate" to format the display.</p> <p>chooser--displays the a chooser button after the text box</p>
Maximum Length	Limits the number of characters that are displayed on toolbar link. The Title property of the button is used as the Alt text to display the full label.	<p>Any number of characters, such as 25</p> <p>When not set, the toolbar displays the entire label.</p>
Image	The image used for the link. This is optional if you define a label, but required if there is no label.	\${COMMON_DIR}/iconSmallOrganization.gif
Input Type	<p>Adds an item to the toolbar. Usually, this setting is used in combination with the Action Label, where the text in the textbox is used as input to the action taken when the button is clicked.</p> <p>If checkbox is specified, only a single checkbox displays with values of Yes or True when checked, and No or False when clear.</p> <p>When used in a structure browser toolbar, only used for tables in Edit mode.</p> <p>Specifies the type of HTML control to display for user input. You can also designate the size of the input boxes to allow for appropriate spacing.</p>	<p>textbox--This is the default. Provides a single-line box for typing text. This graphic shows a textbox and the associated Action Label.</p>  <p>combobox--The input control has a drop-down list of options and users can only select one value. Use combo boxes for attributes that have defined ranges and for attributes whose ranges are determined with a Range Helper URL.</p> <p>submit--Submit button configured to send contents of input control to a processing page.</p> <p>checkbox</p>

	<p>For toolbar commands, textbox, combobox, and submit are supported.</p> <p>For controls other than toolbar commands, submit is not supported.</p>	
Javascript Include	<p>For a menu or command on the toolbar used to specify Javascript files to perform client-side validations. Use this setting in conjunction with the Validate setting on table columns or form fields in place of the application- or component-specific ValidationFile property.</p>	Javascript file name
Popup Modal	<p>If the setting Target Location is set to popup, you can configure the to be modal or non-modal.</p>	<p>true--the popup window is modal false--the popup window is non-modal If not specified, the window is modal.</p>
Range Display Values	<p>A comma-separated list used only for the <code>Input Type = combobox</code> setting. If values for this setting are not provided, the <code>Range Value</code> setting is used to populate the combo box list.</p> <p>The list can be static text or string resource keys.</p>	<p>A,B,C emxFramework.Choice.A,emxFramework.Choice.B,emxFramework.ChoiceC</p>
Range Function	<p>Use to specify the name of the method in the JPO specified in the Range Program setting. See the Range Program setting below for more information.</p> <p>If you specify the Range Program/Range Function and provide a Range Value list, the Range Value list will be used.</p>	<p>The name of a function in the Range Program JPO, such as: <code>getAssignedRange</code> <code>getClassificationRange</code></p>
Range Program	<p>Use to specify the name of a JPO that contains a method to get the ranges (choices) for a combobox input control. A range program is used only when the Input Type setting is combobox.</p>	<p>The name of a JPO, such as: <code>ENCPartFilter</code> <code>AEFUtilToolbarFilter</code></p>
Range Values	<p>A comma-separated list used only for the <code>Input Type = combobox</code> setting. The values for this setting override any Range Function/Range Program value.</p> <p>The list can be static text or string resource keys.</p>	<p>A,B,C emxFramework.Choice.A,emxFramework.Choice.B,emxFramework.ChoiceC</p>
*Registered Suite	<p>The application the command belongs to. The system looks for files related to the link in the registered directory for that application, which is specified in <code>emxSystem.properties</code>.</p> <p>Based on the application name, the system passes the following parameters in the href URL:</p>	<p>The value cannot contain spaces. For example, EngineeringCentral or Framework. The value must be set to the suite name as defined in the key <code>eServiceSuites.DisplayedSuites</code> within <code>emxSystem.properties</code>. If the suite name starts with "eServiceSuite" then this prefix can be skipped and assign the remaining text to the setting. For example, if the suite name in <code>emxSystem.properties</code> is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite".</p> <p>In the href URL that is called when the tree category is clicked, the system passes a parameter called "suiteKey". The value for the parameter is the property name from <code>emxSystem.properties</code> that maps to</p>

	suiteKey emxSuiteDirectory StringResourceId	the setting's value.
Row Select	<p>Used only for links on table pages. Specifies whether the JSP specified for the href expects one, at least one, or no rows in the table (Middle Frame) to be selected.</p> <p>If the Row Select setting is set to "single" or "multi", the setting Submit must be set to "true" to get the details of the selected item(s) from the table.</p> <p>If not specified, it is assumed to be none.</p> <p>A JavaScript alert message is displayed on error conditions when the onClick event occurs. The error conditions are explained in the column to the right.</p>	<p>single-Appropriate only for links that appear on table pages that have radio buttons or check boxes for each row. The JSP specified for the href expects exactly one row in the table (middle frame) to be selected. If more than one item is selected, a JavaScript alert message is displayed. The default message is set to the value of the following key in emxFrameworkStringResource.properties: emxFramework.Common.PleaseSelectOneItemOnly. Currently, the message is set to "Please Select One Item Only". If no item is selected, then the user will see the following message: "Please select an item". This message corresponds to this key value in the emxFrameworkStringResource.properties file: emxFramework.Common.PleaseSelectitem</p> <p>multi-Appropriate only for links that appear on table pages that have check boxes or radio buttons for each row. The JSP specified for the href expects at least one row in the table (middle frame) to be checked. If no item is selected, then the user will see the following message: "Please select an item". This message corresponds to the following key value in the emxFrameworkStringResource.properties file: emxFramework.Common.PleaseSelectitem</p> <p>none (default)-The JSP specified for the href expects no row in the table to be selected. If a row is selected, the JSP does not use it.</p>
Selectable in Preferences	Only for commands in the 3DS (My Desk) submenus. Controls whether the link is available as a preferred Home page. If not set, the system assumes true and the command is listed as a Home page preference. Commands that call pages to display in a popup window, such as an object search command, should not be available as a Home page preference.	True False
Submit	<p>Specifies whether the system should send the object ID(s) (and relationship ID, if applicable) for the current page to the JSP specified in the href parameter. For form pages, the submitted data is just the object ID for the business object the page is about. For table pages, the submitted data is the ID for each selected item in the table, which can include object and relationships. In either case, the IDs are sent using the relBusIdList parameter.</p> <p>If not specified, the system assumes it is false.</p>	true--Submits the IDs to the URL specified in the href parameter. false--The URL specified in the href parameter is directly called from the link and no IDs from the current page are used. Use this value for links that do not require the business object IDs, such as links for creating a new object.
Submit Function	Method within the Submit Program JPO that is executed when a user clicks the submit button for an input control.	getUpdatedList
Submit Program	JPO program that contains the method executed when a user clicks the submit button for an input control.	ENCPartList

Target Location	<p>Controls where the page specified in the href parameter appears.</p> <p>Popup is the default value for a textbox and is used if the setting is not included or if the named frame cannot be found.</p> <p>This value can be set to any valid frame name that is available to the form body frame.</p> <p>When using slidein as the Target Location and the Popup Modal setting is true, then the content window and the global toolbar are grayed out and disabled until the slidein window is closed.</p>	<p>content--The page replaces the content frame.</p> <p>popup--Page appears in new window. Set the modality of the window using the Popup Modal setting.</p> <p>_top--Page replaces the entire body of the browser window.</p> <p>listHidden--Carries out any background processing when no UI is required.</p> <p>hiddenFrame--Targets the Navigator hidden frame to perform any background processing. Do not use this frame within the context of the configurable table. Use listHidden instead.</p> <p>mainFrame--Page appears in the frame that includes the content and menu frames.</p> <p>fromViewHidden--Used within the context of the configurable form.</p> <p>slidein--Page appears in a slidein frame (slides in along the right side of the window).</p> <p>This value can be set to any valid frame name that is available to the form body frame.</p>
width	Specifies the width of the filter toolbar control in pixels. Use this setting if you need to display many controls in a small area.	100
Window Height	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. The window height for the new popup window. Used only when the setting for Target Location is set to "popup".	400 600 (default) 800
Window Width	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. The window width for the new popup window. Used only when the setting for Target Location is set to "popup".	400 600 (default) 800

*Required Setting

Implementing a Toolbar in a JSP

You can include a toolbar in a custom JSP.

1. Configure the required toolbar (administrative menu object) and toolbar items (administrative command objects) and connect them appropriately as described in [Building a Configurable Toolbar](#).

2. Include these style sheets in the JSP page:

- emxUIDefault.css
- emxUIMenu.css
- emxUIToolbar.css

3. Include these JavaScript files in the JSP page:

- emxUIConstants.js
- emxUICore.js
- emxUICoreMenu.js
- emxUIToolbar.js

4. Insert the following section of JSP code into the JSP page.

```
/*
<%
// String sToolBarName = "<Name of the Toolbar Bar - menu object>";
String sToolBarName = "ENCRoutesToolBar";
if ( (sToolBarName!= null))
{
%
<jsp:include page = "emxToolBar.jsp" flush="false">
    <jsp:param name="toolbar" value="<%=>sToolBarName%>" />
    <jsp:param name="objectId" value="<%=> objectId %>" />
?
</jsp:include>
<%
}
%
//*****
```

In the above code, the main parameters to be assigned are:

- sToolBarName: This is JSP page variable assigned to the name of the menu object, which is configured for the toolbar.
- toolbar: This is jsp:include parameter, assigned to the name of the Toolbar menu object.

To support the TipPage parameter, a JavaScript method, openWindow(strURL), must be defined in the JSP page which includes the emxToolbar.jsp.

Searches

You can define a search based on a configurable webform, or you can create a full-text (advanced) search that includes both parameter and text searches.

In this section:

-  [Configurable Search Webform](#)
-  [Full-Text Search](#)

Configurable Search Webform

When adding a search command to a menu, you need to define the configurable search webform and all required parameters.

In this section:

- [About the Configurable Search Webform](#)
- [Settings for emxFormEditDisplay.jsp](#)
- [URL Parameters Accepted by emxFormEditDisplay.jsp](#)

About the Configurable Search Webform

When defining a configurable search webform, you need to understand the structure of the href parameter and how to define a custom toolbar for the search results.

- [Structure of the href Parameter](#)
- [Toolbar for Search Results](#)
- [User Access to Objects](#)

Structure of the href Parameter

The emxFormEditDisplay.jsp allows developers to use a single JSP file for searches.

Currently, each individual search command is configured with an href parameter that calls a custom JSP to execute a search. The search command can be configured with href=emxFormEditDisplay.jsp along with any required parameters.

The search command needs to be attached to the appropriate search menu as described in [Building a Configurable Toolbar](#). To configure the query and results interface to use the configurable table component, the search command must specify:

- Setting: Target Location=searchContent
- The command href=emxFormEditDisplay.jsp
- The URL Parameter program set to the program that processes the search criteria and queries the database
- The URL Parameter table to the table administrative object that defines the layout for the search results page
- The URL Parameter form specifies the webform that defines the search dialog page

When the user clicks **Search** from the Search dialog box, the search component submits the search criteria and all request parameters to the JPO specified in the program URL parameter, and the results are displayed in a table component using the table defined by the table URL parameter.

This sample URL invokes a search for Parts:

```
 ${COMMON_DIR}/emxFormEditDisplay.jsp?program=
emxPart:getPartSearchResult&table=ENCGeneralSearchResult&form=
EngPartSearch&sortColumnName=Name&header=emxEngineeringCentral.
Common.SearchResults&pagination=&Style=dialog&selection=
multiple&HelpMarker=emxhelpfindselectresults&CancelButton=true&
CancelButtonLabel=emxProductCentral.Button.Close
```

See [URL Parameters Accepted by emxFormEditDisplay.jsp](#) for the list of supported and non-supported parameters.

Toolbar for Search Results

If you need to include a different toolbar for the search results, you can pass the resultsToolbar URL parameter.

If you pass the resultsToolbar URL parameter to the webform, you must also add this function to your custom JS file:

```
function doSearch() {
    ....
    ....
    var strToolbar =
currentURL.indexOf ("&resultsToolbar=");
    if (currentURL.indexOf ("&resultsToolbar=") != -1) {
        strToolbarName =
currentURL.substring(strToolbar+16,currentURL.indexOf ("&",strTo
olbar+1));
    }
    theForm.action = ".../common/
emxTable.jsp?toolbar="+strToolbarName;
    ....
    ....
}
```

This code is a portion of the method doSearch() in emxUIFormUtil.js installed by the Business Process Services.

User Access to Objects

Users must have both show and read access to objects in order for those objects to appear in search result lists. When assigning accesses to users, show access without read access performs no function. If only show access is required by a custom application, then a custom search tool must be created.

Settings for emxFormEditDisplay.jsp

The search dialog page, configured using Business Modeler, must invoke `emxFormEditDisplay.jsp` to launch the search results page. This table lists the settings supported by the webform that will invoke `emxFormEditDisplay.jsp`. A list of unsupported settings is included following the table. The list of URL parameters that can be passed to `emxFormEditDisplay.jsp` follows the settings list.

See [Parameters for Web Form Objects](#) for complete descriptions and accepted values/examples.

- [Settings Table](#)
- [Excluded Settings](#)

Settings Table

Setting	Description	Accepted Values/Examples
*Registered Suite	The application the field belongs to.	Set the value without any spaces, for example, EngineeringCentral or Framework.
Access Expression	Only non-object expressions are allowed.	context.user.name == "Test Everything"
Access Function		
Access Program		
Admin Type	Use to translate fields whose values are names of administrative objects or ranges of attributes.	--
Allow Manual Edit	Can user manually edit the field.	false (default) true
Cols	This setting limits the length of the textarea on the form and specifies the visible width in average character widths.	--
Default	Default can be a symbolic name for fields such as Type and Vault.	--
Display Format	Specifies the number of the date format for any field where the value of the format setting is "date."	--
Display Time	Controls whether the time displays with the date for fields whose format is set to date.	true false
Editable	Use to indicate whether the field is displayed as editable or read only. Only applies for Edit mode. View mode ignores the setting.	true (default) false
Field Size	Determines the width of a textbox input type field.	--
Field Type	This setting defines the type of data for the field. The Group Holder setting has been deprecated.	Supported field types: basic attribute Section Header Section Separator program programHTMLOutput Not supported field types: Dynamic Attributes ClassificationPaths ClassificationAttributes image Table Holder Group Holder

		emxTable
format	Use to specify the type of data in the field.	date currency numeric email
function	The name of the method to call within the JPO program specified in the program setting.	--
Group Name	Used for grouping fields in web forms.	The name of the group.
Help Marker	Specifies the name of the help marker to call for context-sensitive help.	--
Hide Label	Displays or hides the label for a particular row on a form.	True False
Input Type	Specifies the type of HTML control to display for user input.	textbox textarea checkbox listbox radiobutton combobox
Label	Use to enter the text that should appear as the label for the field.	--
Maximum Length	Limits the number of characters that can be entered in a field with Input Type of textbox.	--
Popup Modal	Specifies whether windows opened from a link are modal or non-modal.	true false
program	Use to specify the name of the JPO program to get the field's data.	--
RangeHref	Use to configure a textbox that has a Browse (...) button that calls a chooser.	--
Range Function	Use to specify the name of the method in the JPO specified in the Range Program setting.	--
Range Program	Use to specify the name of a JPO that contains a method to get the field value ranges (choices).	--
Remove Range Blank	Used when the <code>input type</code> is set to <code>combobox</code> to ensure that the field contains a value and is not left blank.	true false (default)
Rows	This setting limits the height of the textarea on the form and specifies the number of visible text lines.	--
Section Level	Used in conjunction with the Field Type: Section Header setting to define the level of heading.	1 (default) 2
Show Clear Button	Adds a Clear hyperlink next to the field.	true false
Sort Range Values	Enables or disables the sorting of range values in combobox or listbox controls.	enable (default) disable
Target Location	Controls where the page specified in the href parameter appears when a user clicks the hyperlinked data.	popup content mainFrame <code>_top</code>
TypeAhead	Enables type ahead.	true false

TypeAhead Program	Name of the JPO called to retrieve a list of possible values.	--
TypeAhead Function	The name of the JPO method.	--
TypeAhead Character Count	Overrides the emxFramework.TypeAhead.RunProgram.CharacterCount system property.	2 (default) 5
TypeAhead Saved Values Limit	Overrides the emxFramework.TypeAhead.SavedValuesLimit system property.	10 (default)
Window Height	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. Use to define the height of the popup chooser window or window opened from any href link in a form field.	Number of pixels, such as 700. The default is 600.
Window Width	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. Use to define the width of the popup chooser window or window opened from any href link in a form field.	Number of pixels, such as 700. The default is 600.

*Required Setting



Excluded Settings

Settings not supported by emxFormEditDisplay.jsp:

- Access Mask
- Alternate OID Expression
- Alternate Type Expression
- Category
- Column Count
- Create Exclude
- Edit Exclude
- Field Column Headers
- Field Row Headers
- Field Table Rows
- Field Table Columns
- Group Count
- Image
- Image Size
- Inquiry
- Printer Friendly
- Required
- Show Alternate Icon
- Show type Icon
- sortColumnName
- sortDirection
- table
- Tip Page
- Update Function
- Update Program
- Validate
- Validate Type
- View Exclude

URL Parameters Accepted by emxFormEditDisplay.jsp

This table lists the URL parameters that can be passed to emxFormEditDisplay.jsp.

Parameter	Description	Accepted Input Values
CancelButton	Determines whether or not a Cancel link shows in the results table footer frame. Applies only for tables displayed in popup windows	true false (default)
CancelLabel	Specifies the constant string or string property used as the label for the Cancel link when CancelButton=true.	Default=Cancel
*form	Required by the search interface using the webform. Specifies the web form administrative object used for presenting the form page. You must use the actual form name: symbolic names are not supported.	Name of web form administrative object. <code>emxFormEditDisplay.jsp?form= ENCPart</code>
header	Specifies the constant string or string property used as the header for the search results page.	<code> \${COMMON_DIR}/emxFormEditDisplay.jsp?program= emxPart:getPartSearchResults& table=ENCGeneralSearchResults& form=EngPartSearch&sortColumnName=Name&header=emxEngineeringCentral.Common.SearchResults</code>
HelpMarker	Specifies the name of the help marker to call for context-sensitive help.	<code> \${COMMON_DIR}/emxFormEditDisplay.jsp?program= emxPart:getPartSearchResults& table=ENCGeneralSearchResults& form=EngPartSearch&sortColumnName=Name&header=emxEngineeringCentral.Common.SearchResults&Style=dialog&selection=multiple&HelpMarker=emxhelpfindselectresults</code>
minRequiredChars	The listed fields will use the specified minimum number of characters for wildcard searches; any other fields on the page use the system default set using the <code>emxFramework.FullTextSearch.MinRequiredChars</code> property in <code>emxSystem.properties</code> .	<code>minRequiredChars=TEXT=5:NAME=3:DESCRIPTION=6</code>
pagination	Specifies the number of rows of data to show per page in the search results page. If not used, the system default specified in <code>emxSystem.properties</code> is used. If 0, all objects are displayed in a single page.	0 10 25
*program	Required by the search interface using the webform. Specifies the name of the JPO and the function within the JPO invoked to retrieve the objects from the database based on search criteria.	<code> \${COMMON_DIR}/emxFormEditDisplay.jsp?program= emxProductSearchBase:getUseCases</code>
resultsToolbar	Toolbar to display on the seach results page. You must include the code described in About the Configurable Search Webform in the custom JSP.	Name of menu administrative object that represents a toolbar.
selection	Controls whether or not the results table includes a column of check boxes (supports multiples selections) or radio buttons (supports single selection) as the first colum.	mulitple--displays check boxes single--displays radio boxes none--no selection allowed
showPageURLIcon	When true, shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false, the icon does not show in the toolbar. The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in <code>emxSystem.properties</code> .	true false
sortColumnName	Specifies which column to use to sort table data in the search results page. If not specified, the results page is opens unsorted.	<code> \${COMMON_DIR}/emxFormEditDisplay.jsp?program= emxProductSearchBase:getUseCases& table=PRCSearchUseCasesTable& CommandName=ENCSearchFindParts& sortColumnName=Name</code>
Style	Specifies the style sheet used to display the	list (default)

	header and the results table page. The list value shows the results in the content frame of emxNavigator.jsp. The dialog value indicates a popup dialog box.	dialog
SubmitURL	Determines whether or not the results table includes a Submit link in the footer and provides the name of the JSP. When specified, the selected data is posted to the defined URL when the user clicks the link.	
SubmitLabel	When a value is provided for SubmitURL, the constant string or string property used as the label for the Submit link.	Default=Submit
*table	<p>Required by the search interface using the webform.</p> <p>Specifies the table object used for presenting the search results. Must be the actual table name; symbolic names are not supported.</p>	<code> \${COMMON_DIR}/emxFormEditDisplay.jsp?program=emxProductSearchBase:getUseCases&table=PRCSearchUseCases</code>
toolbar	Specifies the menu administrative object used in the header frame. May use the AEFSearchResultsToolbar as the toolbar to leverage "Revise Search" and "Add to Collection" support	Name of menu administrative object that represents a toolbar.

***Required**

Full-Text Search

You can use the advanced search component, emxFullSearch.jsp (also called the full-text search) to define search pages for your application.

In this section:

- [About emxFullSearch.jsp \(Full-Text Search\)](#)
- [Initial Search Criteria](#)
- [Full-Text Search Results Toolbar](#)
- [Full-Text Search from a Context Page](#)
- [Dynamically-Added Columns to the Results](#)
- [Search Results Sorting Options](#)
- [Form-Based Search](#)
- [Criteria For Real-Time Searches](#)
- [Field Choosers for Full-Text Search](#)
- [Full-Text Search Page Used as a Chooser](#)
- [URL Parameters Passed to emxFullSearch.jsp](#)

About emxFullSearch.jsp (Full-Text Search)

The advanced search component, emxFullSearch.jsp, is a configurable component that defines the full-text search functionality. Full-text search can be implemented by adding a command to the global Search menu, or by configuring the Add Existing functionality for a context object.

For pages configured to use the advanced search component, the emxSystem.properties file includes a property, `emxFramework.FullTextSearch.QueryType`, that determines what type of search is executed when the user selects a search menu item or clicks a browse button in a dialog. If the value for that property is `Real Time`, a consolidated search page (criteria and results on the same page) opens and the advanced search component is not used. See the *Application Exchange Framework User's Guide* for details on the real-time search.

If the property value is `Indexed`, then the advanced search component defined in this section is used. The global property can be overridden for a specific page by including the `queryType` URL parameter. See the *ENOVIA Live Collaboration Installation Guide* for details on installing, configuring, and indexing.

When the property value is `Indexed`, the global toolbar automatically includes a search textbox. You do not need to specifically add the textbox to the toolbar. If the value for this property is `Real Time`, then the search tool on the global toolbar will not be displayed.



When the user clicks the search icon (magnifying glass), the system passes the `txtTextSearch` URL parameter with the user's entered text to emxFullSearch.jsp. The system bypasses the intermediate screen (defined in [Initial Search Criteria](#)) and loads the search results in the navigation or form page as defined by other URL parameters.

Full-text search can be implemented as a navigation-based or form-based structure. In the navigation-based version, the value of the `showInitialResults` URL parameter determines the behavior when the search page is opened. If not defined for the search page, the value of the `emxFramework.FullTextSearch.ShowInitialResults` property is used. If `true`, unfiltered results display when the user opens the search page. If `false`, the user clicks on types, then subtypes, to narrow down the result list, then selects values for any combination of attributes. In the form-based version, the user enters values in form fields to narrow down the result list. For both versions, only attributes that have been indexed are available to the user.

The navigation-based method is the default as defined by the `emxFramework.FullTextSearch.FormBased` property in emxSystem.properties. The default can be overridden using the `viewFormBase` URL parameter passed to a specific search page.

This example shows the navigation-based search with results.

Name	Policy	Rev	Type	Description	State
Acceleration Pedal	EC Part	1	Part	Project 67	Release
Air Valve	EC Part	1	Part	Project 7488	Release
ATPALT0101	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPALT0102	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPALT0103	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPALT0104	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPALT0105	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPBOM0101	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPBOM0102	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPBOM0103	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPBOM0104	EC Part	1	Part	Sample Part for MBOM ATP	Approved
ATPBOM0105	EC Part	1	Part	Sample Part for MBOM ATP	Approved

The example below shows the form-based search with results. The fields shown in a form-based search depend on two things:

- Global fields shown on each form based search window. These fields are defined in the emxSystem.properties file by the property emxFramework.FullTextSearch.FormView.FixedFields. All the search windows tuned for form-based search will show these fields.
- Additional fields per search window. In addition to the global fields, you can show additional fields by passing the url parameter formInclusionList as true.

Full-text search is used in either navigation mode or form-based mode by setting the global property emxFramework.FullTextSearch.FormBased = true/false. This property is found in the emxSystem.properties file. All search windows will use the mode defined in this property. You must restart the web application server after making changes in the emxSystem.properties file.

You can overwrite this global property with the url parameter viewFormBased=true/false. This parameter will be passed like: emxFullSearch.jsp?viewFormBased=true.

Search - Windows Internet Explorer

Type <input type="text" value="Part"/> <input type="button" value="..."/>	Name <input type="text"/> <input type="button" value="..."/>	Revision <input type="text"/> <input type="button" value="..."/> <input checked="" type="radio"/> Highest <input type="radio"/> By State
Owner <input type="text"/> <input type="button" value="..."/>	Policy <input type="text"/> <input type="button" value="..."/>	State <input type="text"/> <input type="button" value="..."/>
Originator <input type="text"/> <input type="button" value="..."/>	Vault <input type="text"/> <input type="button" value="..."/>	Originated <input type="text"/> <input type="button" value="..."/>
Title <input type="text"/> <input type="button" value="..."/>	Mod. fied <input type="text"/> <input type="button" value="..."/>	

Navigation View Collections View

1 - 50 of approx. 1524 Results Page Size: 50 (Max Value: 1000) 1 2 3 4 5 6 7 8 9 10 >

<input type="checkbox"/> Name	Revision	Type	Description	State
Acceleration Pedal	1	Part	Project 67	Released
Air Valve	1	Part	Project 7458	Released
ATPALTO101	1	Part	Sample Part for MBOM ATP	Approved
ATPALTO102	1	Part	Sample Part for MBOM ATP	Approved
ATPALTO103	1	Part	Sample Part for MBOM ATP	Approved
ATPALTO104	1	Part	Sample Part for MBOM ATP	Approved
ATPALTO105	1	Part	Sample Part for MBOM ATP	Approved

The criteria section includes the basics for an object, such as Name, Type, Revision, and so on. Depending on the object type, an additional set of criteria displays for any attributes of that type that have been indexed.

Initial Search Criteria

You need to define the criteria used to show the initial result set for your search page (the results that show when the page is first opened by a user).

The following topics are discussed:

- [Initial Object List](#)
- [Syntax for the field Parameter](#)
- [Revisions](#)
- [Reserved Words](#)

Initial Object List

When the navigation mode is configured to show an initial result set, the initial objects that get loaded are determined by:

- `field` URL parameter (defines initial search criteria for specific fields)
- `default` URL parameter (defines criteria that can be expanded on)
- `txtExcludeOIDs` URL parameter (specifies specific OIDs to omit from the result list)
- `excludeOIDprogram` URL parameter (provides a list of OIDs not to include in the result set based on business logic defined in a JPO program:method)
- `includeOIDprogram` URL parameter (provides a list of OIDS to restrict the search to based on business logic defined in a JPO program:method)

For example:

```
emxFullSearch.jsp?field=TYPE=type_Person:CURRENT=state_Active&default=PROJECT_ID=<projectid>
```

The search page that opens lists Persons who belong to the specified Project.

For cases that need to work with large numbers of OIDs, you can use the `includeOIDprogram` parameter to limit the OIDs being searched and the `excludeOIDprogram` to remove OIDs from being searched.

The field parameter does not limit the vaults to be searched; if you specify a vault, the vault chooser still shows the primary and secondary vaults available to that user.

In addition, see [About Automatic Type Ahead](#) for details on defining the href to support type-ahead fields in the search criteria.



Syntax for the field Parameter

The field parameter specifies the initial search criteria for specific fields in this format:

```
index operator value:index operator value
```

where:

- `index` is the name of the indexed field as defined in the config.xml file (see the "Configuring Advanced Search" chapter in the *ENOVIA Live Collaboration Installation Guide*)
- `operator` is one of these:
 - = the attribute value must equal this value
 - != the attribute value must not equal this value
- `value` is the attribute value to select (or omit) as a match; can be actual value or symbolic names of the value. Values can be a comma-separated list (an OR match, only one of the values need to match)

The list of indexed fields must be separated by a colon (:). If a value includes a space, use single-quotes to enclose the value, not double quotes. For consistency, field names should be provided in upper case, such as OWNER, but must match the name as defined in config.xml.

To specify that the values must match one value but not another value, you can use the = and != operators in sequence. For example, to specify a type (which automatically includes its children) but exclude a specific child type, you can use this format:

```
TYPE=type_Part:Type!=type_HardwarePart
```

In this example, all child types of Part, except for Hardware Part, are considered matches.

The user can then use the Type Hierarchy and Attributes sections or the form to filter the search results as described in the *Application Exchange Framework User's Guide*.



Revisions

When specifying revisions for the initial search criteria, several properties and parameters affect the results. The

emxSystem.properties file includes these properties:

- emxFramework.FullTextSearch.LATESTREVISION = true
- emxFramework.FullTextSearch.LASTREVISION = false

LATESTREVISION corresponds to the **By State** check box; LASTREVISION corresponds to the **Highest** check box. Both properties can be set to `false`, but do not set both properties to `true` (this results in an invalid configuration).

The above properties correspond to these URL parameters that can be passed to emxFullSearch.jsp:

- LATESTREVISION
- LASTREVISION

The URL parameters, if passed, override the emxSystem.properties property values. In addition, only one of these parameters should be passed as true.

You can also pass an explicit Revision level using the field parameter, such as:

`field=REVISION=B`

This example retrieves all objects (that meet the other criteria) that are at Revision level B. If the above URL string also included `LASTREVISION=true`, then the initial results would include all objects that have Revision B as the last revision.

Selected types that are governed by different policies cause complications. For example, if Part and Documents are defined as the Types, these objects have different revision strings and different states. In this case, the Revision text box is used and the check boxes are disabled.



Reserved Words

Do not use any of these reserved words: TYPES, CURRENT, REVISION.

These reserved words do not correspond to indices in the config.xml file. To use them, follow these guidelines:

- TYPES specifies the symbolic name of the types to include in (=) or to exclude from (!=) the result set
- REVISION=last selects the last object in the revision chain
 - `LATESTREVISION=true`- elects the last object in the revision chain in the state specified by the CURRENT keyword (if CURRENT is not passed, functions as REVISION=last)
 - CURRENT defines the policy and state, or just the state

Example:

`TYPES=type_ECR:CURRENT=policy_ECRStandard.state_PlanECO`

In this example, the returned objects must be of type ECR, be governed by the ECR Standard policy and currently in the Plan ECO state.

Full-Text Search Results Toolbar

The full-text search page includes a default toolbar that can be customized.

By default, the search results uses the toolbar that includes these:

- Search in Collection command
- Save Search command
- Generic Delete command
- Printer-friendly icon
- Export icon
- Help icon

You can use the toolbar URL parameter to pass the name of a custom toolbar. See [Menus and Toolbars](#).

Full-Text Search from a Context Page

When initiated from a context page, you may also need to use the submitURL parameter to specify the JSP to handle the user selections.

In addition, you can configure mandatory parameters so that the user cannot de-select that item. For example, if calling the full-text search from a context page with the purpose of adding a specification, you can pass txtType=type_Specification, and mandatorySearchParam=txtType.

Dynamically-Added Columns to the Results

The search results table uses a default table as defined by the table URL parameter (default table is AEEGeneralSearchResults). In addition to columns defined in the table, full-text search may dynamically add columns depending on the user's choices.

Full-text search dynamically adds columns (unless the column already appears in the results set) for attributes if the user selects:

- Multiple values for an attribute
- A date range for an attribute using the On or after, On or before, and between operators
- A number (real or integer) using the <, >, and between operators

If a single value for an attribute is selected, no additional column is added to the results table.

If the search page includes library classifications, the value of the `emxFramework.FullTextSearch.AddDynamicLibraryAttributes` property in emxSystem.properties determines if a column for an attribute in a library class is added to the result set.

You can control where in the table the dynamically-generated column is added. By default, full-text search adds new columns as the first columns after the structure browser freezepane divider. To change this location, use the filterColumnPosition URL parameter (see [URL Parameters Passed to emxFullSearch.jsp](#) for details). This parameter takes a non-negative integer that does not exceed the number of columns, or the word `last`. New columns are inserted at the specified column position, or as the last columns in the table. If an invalid value is passed (such as a negative number or a number that exceeds the number of columns), the value is ignored and the default position (first column after the pane divider) is used.

If the table defined to display the results includes one or more dynamic columns (see [Dynamic Columns](#)), this feature is disabled; additional columns will not be added to the results table based on user attribute selections.

If the user has created a custom view for the results table, then the dynamically-generated column is not added to the results, but it is added to the list of Available Columns for the custom view (unless the column is already in the Visible or Available column list). The user can edit their custom view to add the column to the results table.

Also, if you enable the ability to edit the results (see [Structure Browser Edit Mode](#)), dynamically-added columns cannot be edited by the user.

Search Results Sorting Options

Fast sorting for the search results is controlled by a property in `emxSystem.properties`, the user's pagination setting, and the sort definitions in `config.xml` and `AdvancedSearchIDOLServer.cfg`.

For fastsort to be used in search results, all of these conditions must be met:

- The result count must exceed the threshold set in `emxSystem.properties`:
`emxFramework.FullTextSearch.FastSortThreshold=1000`
- The column selected for sorting must have `fastsort=true` defined on that field in `config.xml`.
- The field must be defined in the `[SetSortFields]` section of the `AdvancedSearchIDOLServer.cfg` file.

If the user has enabled pagination, then fast sort is only used for the specific page. If pagination is disabled, fast sort is executed over the entire result set.

See the *Full-text Search Server Administrator's Guide* for details on the `config.xml` and `AdvancedSearchIDOLServer.cfg` files.

Form-Based Search

You can define the search page to be form-based instead of navigation-based.

When using the form-based search, the type chooser (showChooser call) must be passed all URL parameters passed to emxFullSearch.jsp. For example:

```
showChooser('.../common/emxFullSearch.jsp?callbackFunction=foo&
TYPES=type_Company&table=AEFGeneralSearchResults&selection=sing
le&program=AEFSearchPOC%3Asearch&hideHeader=true&submitURL=.../
common/
emxFullSearchProcess.jsp&suiteKey=Framework&StringResourceFileI
d=
emxFrameworkStringResource&SuiteDirectory=common&objectId=33428
.63973.37136.49473&parentOID=33428.63973.37136.49473&
frameName=pageContent&fieldNameActual=txtCompanyId&fieldNameDis
play= txtProductCompany', 700, 500)
```

Criteria For Real-Time Searches

You can include a webform to present the search criteria for real-time searches for indexed (parametric) object attributes.

You can define a configurable webform to display the search criteria fields when real-time search is used. Real-time search is used if:

- The `emxFramework.FullTextSearch.QueryType` property is set to `Real Time` in `emxSystem.properties`
Or:
 - The `querytype=RealTime` URL parameter is passed to `emxFullText.jsp`.

When defining fields for this form, you must define a `name` and a `label`. Any `select` or `expressiontype` settings defined will be ignored. The field name must be the same as the indexed field definition in `config.xml` in a `<BOTYPEFIELD>` section for the object type being search for. For example, if the `config.xml` file includes these lines:

```
<BOTYPEFIELDS name="person">
  <FIELD name="FIRSTNAME" select="attribute[First Name]"
    type="STRING,PARAMETRIC" />
</BOTYPEFIELDS>
```

You would add a field named `FIRSTNAME` to the webform. In addition, the label for the field should refer to the string resource for that attribute, such as `emxFramework.FirstName`. This example assumes the registered suite is the default, Framework. For fields defined in other products, use the appropriate string resources file.

This table lists the settings supported for fields in the webform.

Setting Name	Description	Possible / Sample Values
Registered Suite	The application the where the string resources are defined for the form labels. The system looks for files related to the component in the registered directory for that application, which is specified in <code>emxSystem.properties</code> .	Framework (default) suitename
Access Expression	Controls access to the field based on a valid MQL expression. The expression gets evaluated at runtime against the context user's Person object. If the expression evaluates to True and no other access control prevents access, the field is shown, otherwise it is hidden.	Any valid expression
Access Function	The name of the JPO method to invoke in the JPO specified for the Access Program setting. The Access function gets the input parameter as a <code>HashMap</code> , which contain all the request parameters that were passed into the Form page. The JPO method must return an object of class <code>Boolean</code> . If the returned value is true and no other access control prevents access, the field is displayed. If false, it is hidden.	JPO method name
Access Program	Controls access to a field based on the output from a method in the specified JPO program. This setting requires that the Access Function setting also be specified. If Access Function is not set, the Access Program setting is ignored.	JPO program name

Field Choosers for Full-Text Search

You can configure fields with choosers so that the user selects field values from the chooser instead of entering text.

When `emxFramework.FullTextSearch.QueryType=Real Time`, you can configure fields in addition to the TYPE and VAULT fields to use choosers. You configure the chooser using the `chooserURL` setting for the field in the config.xml file. In addition, these URL parameters must be passed to the search page:

- `formName=formName` The name of the form that defines the fields for the search criteria
- `fieldNameActual=hidden_FIELDNAME` Name of the hidden field element that stores the actual value submitted
- `fieldNameDisplay=FIELDNAME` Name of the textbox on the form where user's enter search criteria

Business Process Services also provides pre-defined type ahead choosers for Type, Organization, or Persons. See [Automatic Type Ahead](#).

To provide a set of range values instead of a chooser for a field, you can use the `chooserJPO` setting for the field in the config.xml file to specify the JPO:Method.

In an indexed search, non-parametric fields configured with choosers (chooserURL or chooserJPO configured for the field in the config.xml file), can use current field values from the original search page in fields in the search page opened by clicking a chooser button.

The URL parameters pass current values from the original search page to the page opened by the chooser:

`default=FIELDNAME=${FIELDNAME}`

For example, the Division field could be defined in config.xml with a chooser like this:

```
<FIELD name="DIVISION" chooserURL="${COMMON_DIR}/  
emxFullSearch.jsp?default=NAME=${NAME}:POLICY=${POLICY}"  
select="division" type="STRING" />
```

This definition passes the current values for the Name and Policy fields as default values for those fields in the opened search page. If the search result using these values is a single object, then that object is populated in the field without opening and executing the search.

See the *ENOVIA Live Collaboration Installation Guide* for details on editing the config.xml file.

Full-Text Search Page Used as a Chooser

When defining a field to use a chooser for data entry, that chooser function can also be a full-text search page.

When defining a chooser using `chooserURL` as described in [Field Choosers for Full-Text Search](#), you can specify `emxFullSearch.jsp` as the chooser. In this case, when the user clicks the ellipsis button, another full-text search page opens. You can use macros to configure the chooser to pass any value entered in that field in the original full text search page to the chooser search page. For example, the Title field can be defined like this:

```
emxFullSearch.jsp?field=NAME=${TITLE}
```

If a user enters a value in the Title field of the search form, then clicks the chooser button, that value is passed to the chooser and pre-populated in the Name field.

If the criteria passed to a full-text-as-chooser search page results in a single object, the second search page is not opened and that object is immediately entered as the criteria for that field.

URL Parameters Passed to emxFullSearch.jsp

In addition to the parameters defined in this section, you also need to pass any URL parameters required for the Structure Browser.

The submitURL and header URL parameters should not be passed; these functions are defined within emxFullSearch.jsp. In addition, you should not pass the program, expandProgram, or relationship URL parameters (no expansion of nodes is supported in the search results).

See [Structure Browser Edit Mode](#).

Parameter	Description	Accepted Input Value/Example
cancelLabel	<p>Shows a Cancel button that closes the pop-up window without any other actions. By default, the label is "Cancel".</p> <p>Can be a text string or a string resource id.</p>	emxFramework.GlobalSearch.Done = Cancel emxFramework.GlobalSearch.Select=Close
default	For each field in the form-based view, you can pass default values. The search page uses these values to load the initial search results, and users can change these values as needed.	default=POLICY=policy_ECPart,policy_DevelopmentPart
enableSearchButton	<p>The <code>emxFramework.FullTextSearch.EnableSearchButton</code> property defines globally whether the Search button is enabled prior to the user entering criteria.</p> <p>This parameter overrides that property for the specific page. The default value for this parameter is the value set for this property in <code>emxSystem.properties</code>.</p>	true false
excludeOIDprogram	The JPO program:method that returns a list of Object IDs (OIDs) that should not display in the result list. This program:method provides an additional level of filtering.	JPO program:name emxECR:getAffectedItemsAlreadyConnected
field	A colon-separated list of indices and the default/allowed values for the initial search results. See Initial Search Criteria . In the value for this parameter, colons (:) act as the AND operator and double-pipes () act as the OR operator.	Type=type_Part:Originator='Test Everything':Policy=policy_P1,policy_P2:LatestRevision=true TYPES=Person:LASTNAME=*William" FIRSTNAME=*William" USERNAME=*William*
fieldLabels	Allows for custom fields names. The value for this parameter can be an absolute value or a string resource, and you can provide a comma-separated list of values.	fieldLabels=NAME:emxFramework.Document.Number,DESCRIPTION:Custom Description Name
fieldNameActual	Optional. The name of the field where the selected value from the search results is returned.	<string>
fieldNameDisplay	Optional. If fieldNameActual is used and the actual value and display value are different, this parameter defines the name of the field where the display value should be returned.	<string>
filterColumnPosition	<p>Optional. Defines where dynamically-generated columns based on a user's attribute selection will be inserted in the results table.</p> <p>The value must be non-negative and cannot exceed the number of columns in the results table. Or, you can use last to indicate that the column(s) should be inserted after the last column.</p> <p>If no parameter is passed, the columns are inserted after the structure browser pane divider.</p>	A positive integer or the reserved word <code>last</code> (case insensitive)
fieldSeparator	<p>Defines the character to use to separate fields in the URL string. This parameter overrides the value set using the <code>emxFramework.FullTextSearch.FieldSeparator</code> property defined in <code>emxSystem.properties</code>.</p> <p>Do not use any of these values:</p>	^sep^

	<p>value of the <code>emxFramework.FullTextSearch.RefinementSeparator</code> property (these 2 properties must have different values)</p> <pre># ! & % (any text that begins with the percent symbol, such as %3A, %3D, %2C) [or] {or } ? ' "</pre>	
form	The name of the webform used to define the search criteria for real time searches.	formName
formInclusionList	Optional. Comma-separated list to limit the list of indexed attributes to display on a form-based search. This parameter does not affect navigation-based searches.	Name,Description,Policy,attribute_title
frameName	Optional. If using form-based search, this parameter provides the name of the frame that contains the form.	<string>
genericDelete	When true, shows the Delete action (AEFGenericDelete command) on the toolbar for the search page. When false, the Delete action is not included in the toolbar.	true false (default)
hideHeader	The header includes the search text box, and the Search and Reset buttons. For full-text search, the default is true (although for other structure browser pages the default is false).	true (default) false
hideToolbar	Shows or hides the toolbar in the search results section of the search page. To hide the toolbar, set this parameter to true.	true false (default)
includeOIDprogram	The JPO program:method that returns a list of Object IDs (OIDs) to search within. This program:method provides an additional level of filtering.	JPO program:name emxECR:getAffectedItems
LASTREVISION	When true, the Highest check box for the Revision field is selected when the page opens. When false, the check box is clear. Default value is the value for the <code>emxFramework.FullTextSearch.LASTREVISION</code> property in <code>emxSystem.properties</code> . Passing this URL parameter overrides the default value set in <code>emxSystem.properties</code> .	Important: Do not set both this parameter and the LASTREVISION parameter to true.
LATESTREVISION	When true, the By State check box for the Revision field is selected when the page opens. When false, the check box is clear. Default value is the value for the <code>emxFramework.FullTextSearch.LATESTREVISION</code> property in <code>emxSystem.properties</code> . Passing this URL parameter overrides the default value set in <code>emxSystem.properties</code> .	Important: Do not set both this parameter and the LASTREVISION parameter to true.
mandatorySearchParams	A comma-separated list that defines mandatory search	Type

	<p>criteria. The user will not be able to de-select any criteria specified.</p> <p>Real Time mode does not support this parameter.</p>	Originator, Policy
pageSize	An integer defining the initial page size (number of rows) for the search results. The parameter overrides the emxFramework.FullTextSearch.PageSize property.	50 500
queryLimit	For the specific full-text search page, overrides the limit defined by the emxFramework.FullTextSearch.QueryLimit property in emxSystem.properties for the maximum number of results returned from a search	<integer value>
queryType	Default value is defined by the emxFramework.FullTextSearch.QueryType property in emxSystem.properties.	Real Time Indexed
searchCollectionEnabled	<p>When true, includes the Search in Collection toolbar button for navigation-based searches.</p> <p>This parameter is not recommended when searching type-specific tables, or results tables that use expandJPOs or relationships.</p>	true (default) false
selection	Controls whether the table page adds a column of check boxes or radio buttons in the left-most column of the search results.	<p>multiple--Default. Users can select more than one row in the table. A check box displays in the left column of each row. There are no access restriction for the check boxes.</p> <p>single--Users can select one row in the table. A radio button displays in the left column of each row.</p> <p>none--A selection column is not added to the table page.</p>
showInitialResults	<p>If true, shows an unfiltered result set when the full-text search (navigation mode) is opened. If false, the user must select initial criteria and click Search and then a result set is displayed.</p> <p>This value overrides the system property <code>emxFramework.FullTextSearch.ShowInitialResults</code> set in emxSystem.properties.</p>	true false
showPageURLIcon	<p>When true,  shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false, ICON does not show in the toolbar.</p> <p>The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in emxSystem.properties.</p>	true false
showSavedQuery	Includes the AEFSaveQuery command on the page toolbar.	true (default) false
submitLabel	The label to display on the submit button. If not passed, the label shows as "Done". You can pass a static text string or a string resource value (recommended).	emxFramework.GlobalSearch.Done = Done emxFramework.GlobalSearch.Select=Select
SubmitURL	Defines a callback URL. The JSP called when a user clicks the Submit button. For example, the page that performs post-processing when the search was initiated from an Add Existing command.	../engineeringcentral/emxFullSearchAddPartEC.jsp
table	Defines the name of the table used to display the search results. By default, the AEFGeneralSearchResults table is used.	AEFGeneralSearchResults (default) ENCPartSearchResults
toolbar	Defines the name of a custom toolbar to display in the search results frame. Use this parameter only if you need to define a custom toolbar for context search using the consolidated search component.	string
txtExcludeOIDs	Comma-separated list of Object IDs (OIDs) to exclude from the search results.	
txtTextSearch	If a value is passed using this parameter, that value is	string

	used as a search criteria for the initial load of the search page. The search criteria/results page loads with the results of this initial search, and the user can continue entering criteria as needed.	
viewFormBased	Overrides the emxFramework.FullTextSearch.FormBased property defined in emxSystem.properties. True uses the form-based search; false uses the navigation-based search.	true false (default)

Navigation Trees

You can customize navigation trees for your application. Navigation trees present information related to a particular object and display as the Categories menu for the context object.

In this section:

- [About Navigation Trees](#)
- [Structure Navigator](#)
- [Definition of a Navigation Tree](#)
- [Sub-trees Configured as Categories](#)
- [Dynamic Categories for a Tree](#)
- [Structure Tree for an Object](#)
- [Navigation Tree with Revision Filters](#)
- [Navigation Trees Linked to an Application](#)
- [Building a Navigation Tree](#)
- [Tree JavaScript APIs for Custom JSPs](#)
- [Deprecated Methods \(Navigation Tree\)](#)
- [Parameters for Tree Menu Objects](#)
- [Settings for Tree Menu Objects](#)
- [Parameters for Tree Category Command Objects](#)
- [Settings for Tree Category Command Objects](#)
- [URL Parameters Accepted by emxTree.jsp](#)
- [About the Default Tree](#)
- [Specific Trees for an Object Type](#)
- [About the Icon for a Type's Tree](#)
- [Dynamic Expand for Trees](#)
- [Guidelines for Writing Structure Tree JPO](#)
- [Accessing Applications Externally](#)

About Navigation Trees

Navigation trees let users access information related to a particular business object.

Navigation trees consist of:

- root node that represents the business object
- categories under the root node that represent information related to the business object, such as history, attached files, and approvals

The following graphic shows an example of the Categories menu, the implementation of the navigation tree, that the system displays when a user views detailed information for a part. The root node containing the name of the object always displays at the top of the tree. The categories of relevant information display in a menu list below the object name.



Users view the information in a tree by clicking a particular category. Most categories display table pages with lists of objects in the category for the selected object. To view the Properties page for the object, the user clicks the object's name at the top of the tree or the Properties category. By default, the system automatically selects the object's name when the tree first opens, but you can configure a different category as the default display. (See the DefaultCategory parameter for emxTree.jsp and the Default Category setting for tree menu objects).

Because different object types typically require different categories, you can configure trees for each type of object. For example, the tree for a part needs a category for related ECRs, while the tree for a supplier needs a category for locations. Trees for different object types may have categories in common, such as history and attached files so Business Process Services provides a default tree containing many common categories. The system uses this default tree if you do not define one for an object type. You can control user access to every category in a tree. For more information, see [About the Default Tree](#).

Users access a tree by choosing a business object from a table page. (Within the context of the user interface design, this is the only way for users to select a tree, but a different design could allow users to access trees from other components.) The following graphic illustrates a user selection.

The screenshot displays two windows from a software application. The top window is titled 'Features' and contains a table with three rows. The columns are labeled 'Name', 'Revision', 'Type', 'Description', and 'State'. The first row has a selected item 'F1' highlighted with a blue border. The second row has 'T1' and the third row has 'J2'. The bottom window is titled 'Feature F1 rev A: Properties' and shows a navigation tree on the left under the heading 'F1 A'. The tree includes categories like Assignees, Configuration Rules, Compatibility Rules, Resources, Rule extensions, Engineering Changes, UGOM, History, Images, Issues, Lifecycle, Product Configurations, Quantity Rules, Reference Documents, Revisions, Routes, Specifications, Subfeatures, Test Cases, and Where Used. To the right of the tree, there is a section titled 'F1 A (continued)' which lists Derived Features and Manufacturing Plans.

The system displays the object's navigation tree as the Categories menu. By default, the system selects the default page for the object.

Within the context of the dynamic UI design, categories in navigation trees are always nouns. They present lists of related items or attributes for the item. Action commands are in the page Actions menu.

Structure Navigator

Certain types of objects support hierarchical structures. For example, projects in Program Central are hierarchical: they have folders, which can contain subfolders. Each subfolder can contain subfolders and so on. You can configure these objects to display in a structure navigator.

A structure navigator is a frame that displays the structure of the context object. The structure navigator contains only items within the defined hierarchy and displays only their names, allowing users to quickly navigate to the pertinent items in the hierarchy.

The screenshot shows a software interface with a sidebar on the left containing a tree view of a 'DocSpace' structure. The tree includes nodes for 'Results(2)', 'Specifications(4)', and 'Templates(5)', with 'Word Templates(1)' being expanded to show 'Word Template(1)'. The main area is titled 'Specifications : Content' and displays a table with four rows, each representing a document template. The columns are labeled 'Actions', '0/1', and 'Name'. The names listed are 'DOC-0000006', 'DOC-0000007', 'DOC-0000008', and 'DOC-0000009'. The row for 'DOC-0000007' is highlighted with a light blue background.

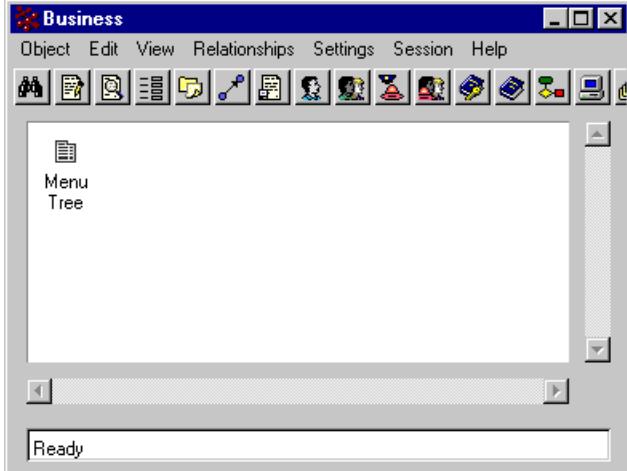
Actions	0/1	Name
	0/1	DOC-0000006
	0/1	DOC-0000007
	0/1	DOC-0000008
	0/1	DOC-0000009

By default, the structure navigator opens when a user views details for an object that you have configured as a structured object.

For instructions on configuring an object so its structure displays in the structure navigator, see [Structure Tree for an Object](#).

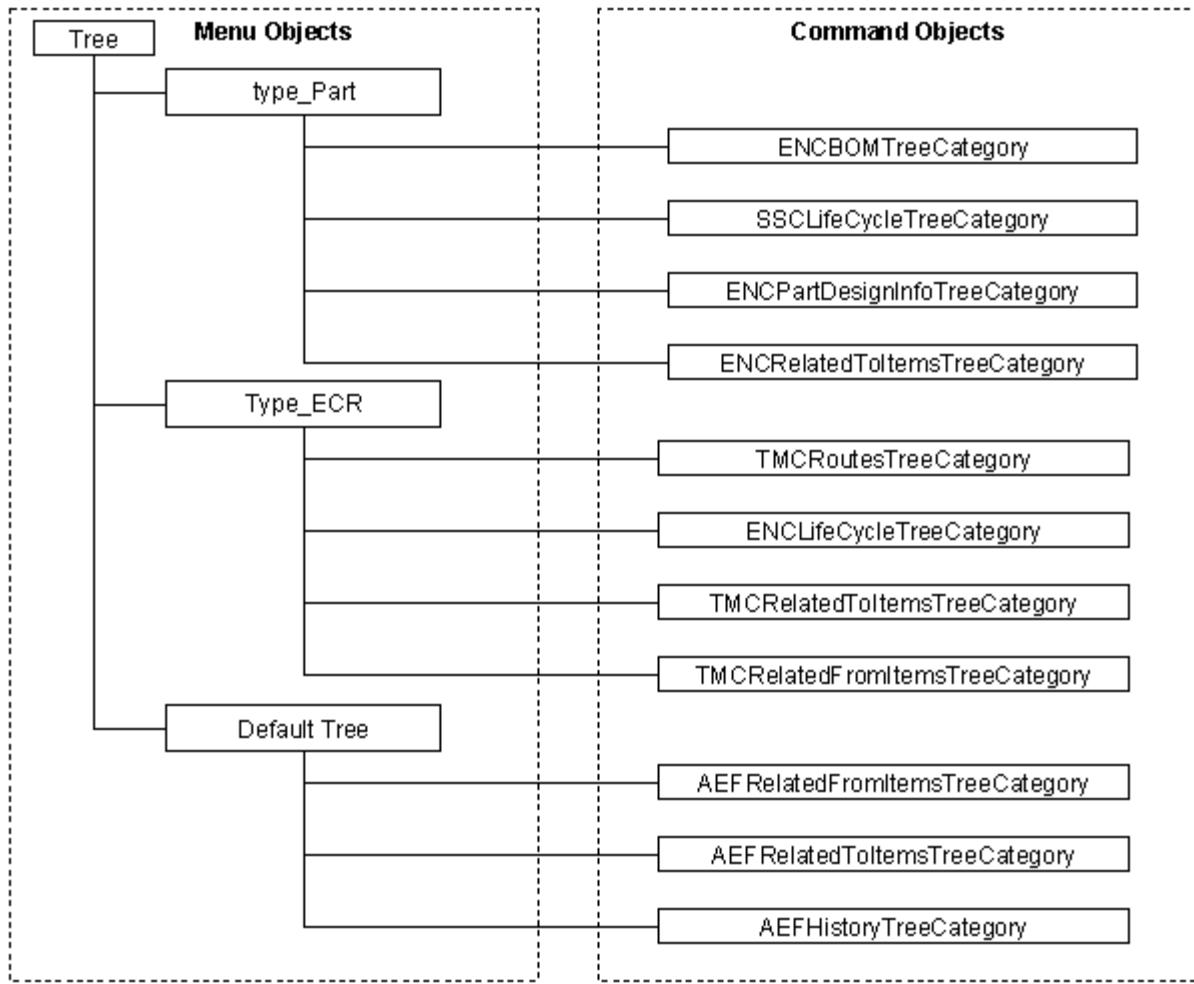
Definition of a Navigation Tree

Business Process Services installs an administrative menu object, called Tree, which represents the navigation trees in the dynamic UI, used to define individual trees.



The definition for the Tree menu object includes a menu object that represents the tree for every object type that needs a navigation tree. For example, Engineering Central installs a menu object for parts and one for ECRs. These menu objects are assigned to the Tree menu object. The menu object for each navigation tree includes a command object for each category in the tree.

This graphic shows a sample administrative object structure for an installation that needs trees for parts, ECRs, and the default tree.



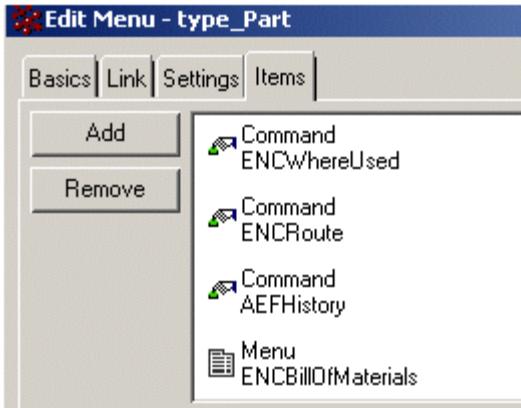
A menu with the symbolic name of an object type, such as type_ECR, defines the navigation tree for that object type. The navigation tree displays in the UI as the Categories menu for the context object.

Business Process Services installs a menu object called Default Tree, which is assigned to the Tree menu object. The system uses the default tree when there is no tree defined for a specific object type or any of its parent types. For information on configuring the default tree, see [About the Default Tree](#).

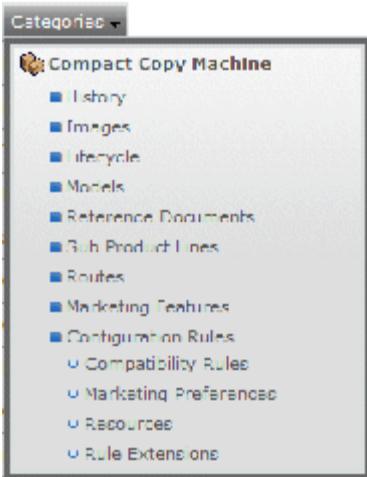
Sub-trees Configured as Categories

You can configure a tree's category as a sub-tree by defining the sub-tree in Business Modeler as you would any tree: create a menu object and assign commands to it.

You assign menus to menus in the Items tab, where you assign commands to a menu.



Here is an example of a Categories tree for a Product that contains a sub-tree, the Configuration Rules category.



Business Process Services automatically displays all submenus to all levels when the user opens the Categories menu.

Dynamic Categories for a Tree

You can write a JPO that dynamically adds categories to a tree.

The following topics are discussed:

- [About Settings for Dynamic Columns](#)
- [Structure of the Returned HashMap](#)
- [Sample JPO for Obtaining Dynamic Categories](#)

About Settings for Dynamic Columns

The JPO is configured with the tree, or a menu or command in the tree, using the Dynamic Command Program and Dynamic Command Function settings.

These settings can only be used on a "static" menu or command; they cannot be used with menus or commands added dynamically. That is, you cannot nest dynamic categories.

When a tree is loaded, the system invokes any JPO configured by these settings on the tree or any command in the tree. If defined for a menu in the tree, the JPO is invoked when the user expands that category. The JPO returns a hashMap containing key/value pairs to generate a dynamic category. The hashMap can contain any number of key/value pairs as required by the business logic. The JPO must implement the business logic that determines which dynamic commands/menus should be included.

If these settings are defined on the tree menu itself, the returned categories (menus or commands) are included at the top of the category list followed by the static (specifically-defined) categories.

If the settings are defined on a command within the tree's structure, the returned categories take that command's place in the tree structure.

If the settings are defined on a menu within the tree's structure, the JPO is not invoked until the user expands that category, and the dynamic categories for the menu are added as subcategories for the menu category along with any defined static categories.



Structure of the Returned HashMap

The hashMap returned to the application from the JPO must contain key/value pairs that define the categories to add to the tree, either in addition to the existing static categories (when configured on the tree), in place of an existing command (when configured for a command in the tree), or as subcategories for a menu (when configured for a menu in the tree).

This table defines the contents of the hashMap:

Key	Description	Sample Value												
name	Name of the category	Dynamic Property												
label	The text to display in the tree.	Dynamic Property												
href	The URL to execute when the user clicks the category.	<code> \${COMMON_DIR}/emxTable.jsp \${SUITE_DIR}/ emxpartEditPartDialog.jsp</code>												
type	Defines if the hashMap is returning a command or a menu to add as a category.	command menu												
roles	If the type is a command, this key defines the roles that have accessed to the returned category.	<code>new StringList("all")</code>												
settings	A hashMap that defines the settings to apply to the command/menu to be inserted as a category. See Settings for Tree Menu Objects for a list of supported settings and their descriptions. The table to the right shows typical settings.	<table border="1"><thead><tr><th>Key</th><th>Sample Value</th></tr></thead><tbody><tr><td>Registered Suite</td><td>Framework</td></tr><tr><td>Target Location</td><td>content</td></tr><tr><td>Access Expression</td><td><code>context.user.isAssigned[ProjectUser] == TRUE</code></td></tr><tr><td>Access Program</td><td>emxUIAccessUtil</td></tr><tr><td>Access Function</td><td><code>getObjectAccess</code></td></tr></tbody></table>	Key	Sample Value	Registered Suite	Framework	Target Location	content	Access Expression	<code>context.user.isAssigned[ProjectUser] == TRUE</code>	Access Program	emxUIAccessUtil	Access Function	<code>getObjectAccess</code>
Key	Sample Value													
Registered Suite	Framework													
Target Location	content													
Access Expression	<code>context.user.isAssigned[ProjectUser] == TRUE</code>													
Access Program	emxUIAccessUtil													
Access Function	<code>getObjectAccess</code>													

children	If the type is menu, this key is a mapList that contains another hashMap as defined in this table for each of the child categories to add.
----------	--



Sample JPO for Obtaining Dynamic Categories

This sample JPO shows how to obtain dynamic categories for a tree.

```
public HashMap getDynamicCategories (Context context, String[] args )throws Exception
{
    HashMap resultMap = new HashMap();
    MapList categoryMapList = new MapList();
    /* Unpack the Arguments to get the Input param Map */
    HashMap inputMap = (HashMap)JPO.unpackArgs(args);
    /* Input param Map has paramMap and commandMap
     * paramMap contains objectId, relId, mode, treeMenu,
     requestMap */
    HashMap paramMap = (HashMap) inputMap.get("paramMap");
    /* commandMap retrieved form Input param Map
     * The static command/menu configured with Dynamic Settings
    */
    HashMap commandMap = (HashMap) inputMap.get("commandMap");
    /* requestMap is retrieved from paramMap. requestMap
contains
 * request parameters passed to the Tree Component */
    HashMap requestMap = (HashMap) paramMap.get("requestMap");
    String strObjId = (String) paramMap.get("objectId");
    /* Creation of a Dynamic Command Category starts...*/
    /* New Dynamic Command categoryMap is created*/
    HashMap categoryCommandMap = new HashMap();
    /* Dynamic CategoryMap should have value for type, name,
settings,
     * roles, href. Otherwise that particular category is
rejected in
     * the result */
    /* type should be command for command
     * category type should be menu for menu category
     */
    categoryCommandMap.put("type", "command");
    /* label can be property entry or a hard coded value */
    categoryCommandMap.put("label", "emxFramework.DynamicCommand.
propertykey");
    /* roles should be StringList, not String, and can have
multiple
     * values
     * Ex: StringList roles = new StringList(); roles.add("Public
Add");
    roles.add("Product Management"); */
    categoryCommandMap.put("roles", new StringList("all"));
    /* name should not be null and it should be unique in the
current
     * tree (i.e.: should not match any existing category name in
the
     * view) */
    categoryCommandMap.put("name", "DynamicCommand");
    /* href which is executed when this category is click by the
user
     * can be blank for menu category (i.e.: type = menu) */
    categoryCommandMap.put("href", "${COMMON_DIR}/
emxDynamicAttributes.jsp?objectId="+strObjId);
    /* settings for the category item All the setting which are
     * applicable to static
     * category can be used. It should be a separate HashMap */
    HashMap settings = new HashMap();
    settings.put("Registered Suite", "Framework");
    settings.put("Target Location", "content");
    settings.put("Access Expression", "context.user.name==\"Test
")
}
```

```

Everything\ "");

    /* settings is added to the category Map settings can be null
 */
    categoryCommandMap.put("settings",settings);
    /* Creation of a Dynamic Command Category ends...*/

    /* Creation of a Dynamic Menu Category starts...*/

    HashMap categoryMenuMap = new HashMap();
    categoryMenuMap.put("type", "menu");
    categoryMenuMap.put("label", "emxFramework.DynamicMenu.
propertykey");
    categoryMenuMap.put("description", "dynamic menu");
    categoryMenuMap.put("roles", new StringList("all"));
    categoryMenuMap.put("name", "DynamicMenu");
    categoryMenuMap.put("href","");
    categoryMenuMap.put("settings",null);

    HashMap grandChild = (HashMap)categoryCommandMap.clone();
    grandChild.put("name", "DynamicGrandCommand");
    grandChild.put("label", "Dynamic Grand Command");
    MapList menusSubCategories = new MapList();
    menusSubCategories.add(grandChild);

    /* Note: when the category is menu it should have all its Sub
     * Categories in a MapList and the key should be Children if
     * the key
     * Children is not there Menu Category is rejected */
    categoryMenuMap.put("Children",menusSubCategories);

    /* Creation of a Dynamic Menu Category ends...*/
    /* categories are added to the categoryMapList */
    categoryMapList.add(categoryCommandMap);
    categoryMapList.add(categoryMenuMap);

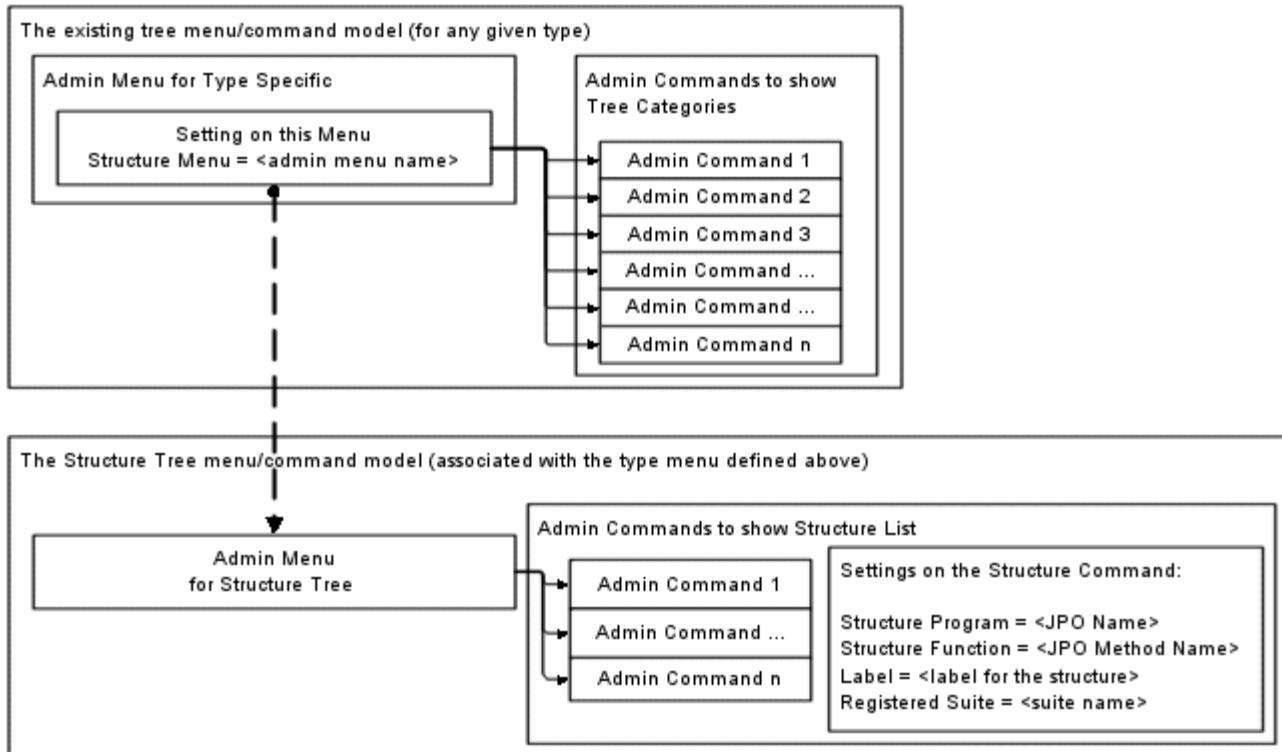
    /* resultMap should have a key Children and value should be a
     * MapList containing CategoryMaps(s). */
    resultMap.put("Children",categoryMapList);
    return resultMap;
}

```

Structure Tree for an Object

This section describes how to configure an object type so the objects related to it in a hierarchical structure display in the Structure Navigator.

The model below shows the two administrative menus that need to be defined for implementing an object's structure tree.

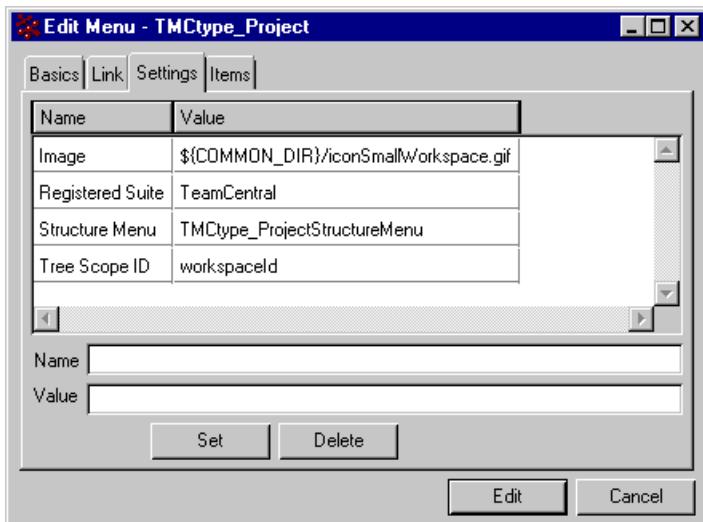


The top menu is the standard tree menu for any object type. All the applications use this tree menu to show the tree component. You define this menu for a spesific object type to show the root node and its categories.

The second structure menu is associated with the regular menu and displays a structure tree along with the regular tree. You associate this second administrative menu to the main regular tree menu using the following setting:

Structure Menu = <the admin menu name to be used for Structure>

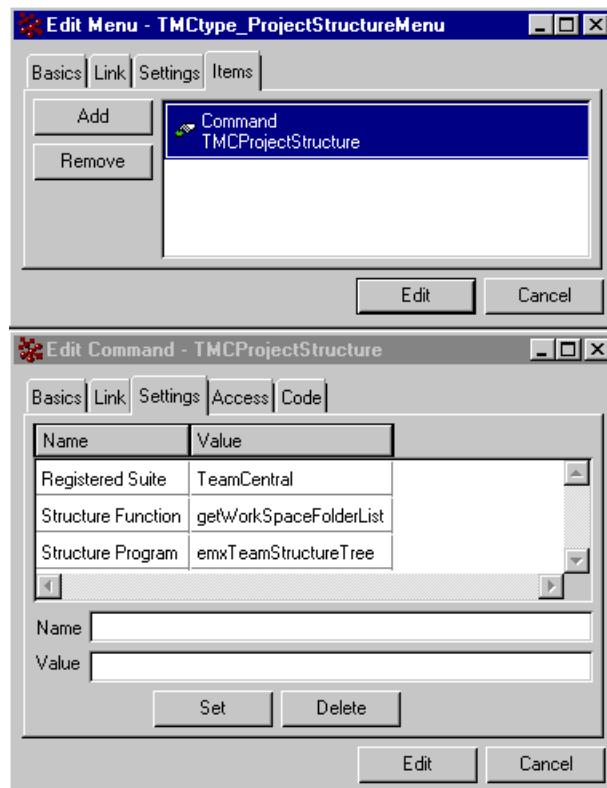
The following examples shows the menu object for the workspace tree in Team Central.



The administrative menu that represents the structure tree is connected to one or more administrative command objects. Each connected command has the necessary settings to generate a specific structure for that object. The settings define an inquiry or JPO that gets a list of business objects to display. If more than one command is connected to the structure menu, the Structure Selection combo list displays on the Structure Navigator to let users choose the structure they want to see (see

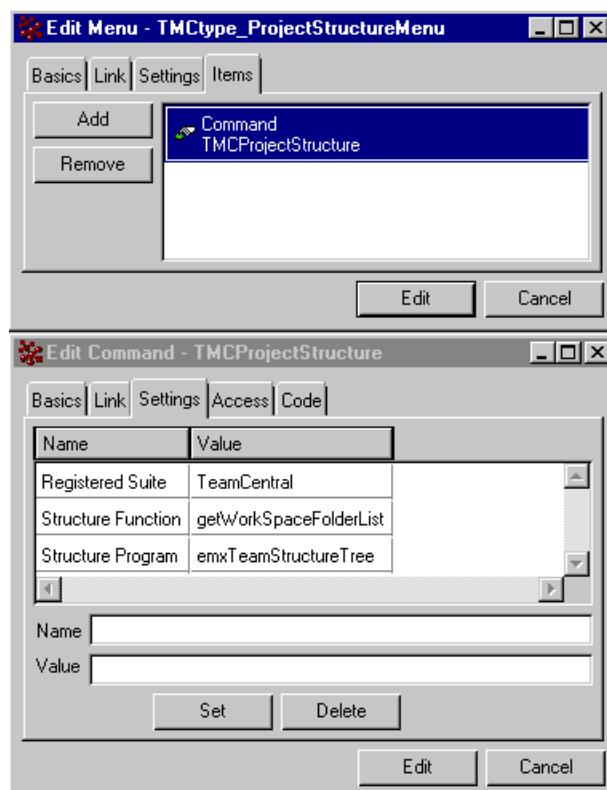
[Structure Navigator](#)).

This is the structure menu for Team Central workspaces. The only structured objects for workspaces is folders, so only one command is connected to the menu.



Using settings on the structure command objects, you can configure the label for the combo box, and the JPO or inquiry to use for retrieving the object list when users navigate through the structure. For descriptions of the settings, see [Settings for Tree Menu Objects](#).

This graphic shows the settings for the command connected to the structure menu for Team Central workspaces.

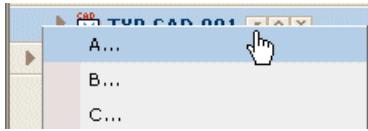


Navigation Tree with Revision Filters

Object names in the navigation tree include the revision level. You can specify a type revision filter on a Type's tree menu using the Revision Filter=Enable setting.

If the object has more than 1 revision, this setting adds an *r* button after the object name in the tree.

If the object has a single revision, the button does not show. Clicking the *r* opens a drop-down menu listing all revisions of that object.



When a user clicks a revision, a new window opens showing the Properties page for the selected revision.

Navigation Trees Linked to an Application

In the dynamic user interface, users access navigation trees by selecting a business object from a table page--usually either the column that contains the object name or a popup details column. The JSP that constructs a tree must call the configurable JSP for trees, emxTree.jsp, plus any additional parameters.

See [URL Parameters Accepted by emxTree.jsp](#).

emxTree.jsp is the main JSP to control the display of any tree, and it is located in ematrix/common. You can create the new tree in a separate popup window or replace the existing page.

By default, the emxTree.jsp gets the menu object that has the same symbolic name as the selected object's type and is connected to the top-level tree menu called Tree. emxTree.jsp requires the ID for the selected object. You pass this information using the objectId parameter. For example, if the selected object is a part, the page looks for a menu object named type_Part. If no such object exists, the page uses the tree defined for the object's parent type. If there is no tree for the parent, the page looks for a tree defined for the grandparent, and so on. If the page reaches the top level type and finds no tree defined for it, it uses the default tree. (For information on using application-specific alternate trees and custom trees for an object, see [Specific Trees for an Object Type](#).)

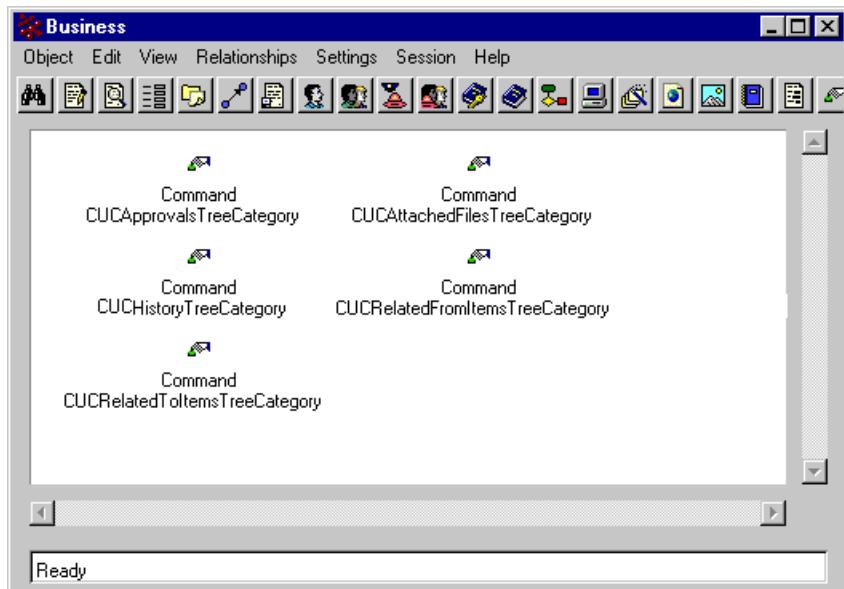
Building a Navigation Tree

This procedure lists the main steps for creating a tree and adding it to an application.

1. Create a command object for every category you want to include in the tree. You can use command objects in multiple menus and trees. If you have already created a command for the category, you do not have to create another command object. You need to define these items for the command:

- The page to display when a user clicks the tree category
- How the page should appear
- Who can access the page

For details about the available parameters and settings for tree category commands, see [Parameters for Tree Category Command Objects](#). For naming conventions, see [Naming Conventions](#).

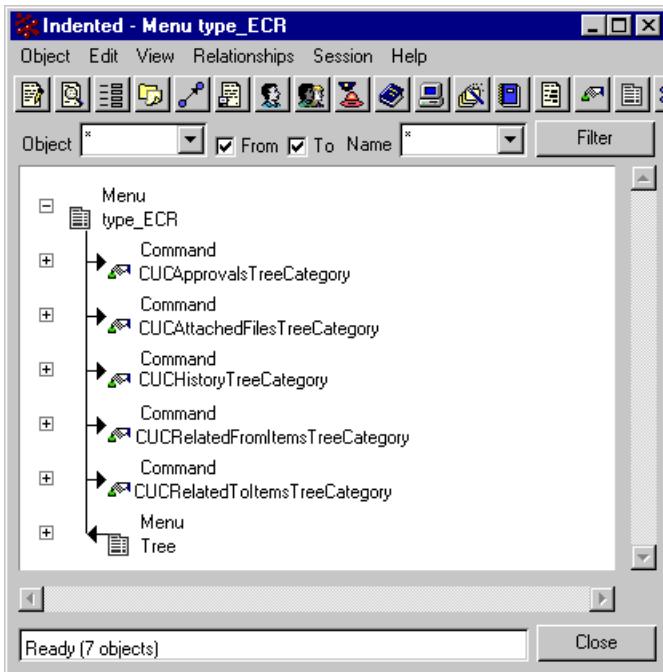


2. Create a menu object for the tree and make sure the object's name is the symbolic name for the tree type. For example, the tree for parts is named type_Part. Although not recommended, you can use the treeMenu parameter for emxTree.jsp to create a custom tree menu for an object type and override the standard tree menu. For object types used in multiple applications (such as Part, Buyer Desk, Person), use alternate trees. For more information, see [Specific Trees for an Object Type](#).)

When you create a menu object for a tree, you define how the tree should appear, and the categories it should include. For a description of how to fill in the parameters and settings for tree menu objects, see [Parameters for Tree Category Command Objects](#).

3. Assign the command objects for the tree categories to the menu object for the tree. In Business Modeler, the order the commands are listed in the Items tab of the Create or Edit Menu dialog box are the order they appear in the tree. To change the order, use drag-and-drop.
4. To assign a sub-tree within the tree, assign the menu object that represents the sub-tree to tree menu object.
5. Connect the menu object for the tree you created in Step 2 to the Tree menu object installed with Business Process Services.

This graphic shows the menu object for ECRs with the connected command objects for categories and the connected Tree menu object.



6. Make sure the tree's object type has an icon defined in emxSystem.properties (or you can use the default icon.) For information, see [About the Icon for a Type's Tree](#).
7. Call emxTree.jsp and specify the parameters to display the tree and associated page. Here are some example href values for updating a tree from a JSP. For a description of the parameters, see [URL Parameters Accepted by emxTree.jsp](#).

```

<a href=". /common/emxTree.jsp?objectId
=<%=sPartId%>&mode=insert&jsTreeID
=<%=jsTreeID%>" class="object" target="content">><%=sPartName%></a>

<a href=". /common/emxTree.jsp?objectId
=<%=OID%>&mode=replace&jsTreeID=<%= jsTreeID%>" class="object"
target="content">><%=sRTSname%></a>

<a href="common/emxTree.jsp?objectId
=<%=sPartId%>&mode=insert&jsTreeID=" class="object"
target="content">><%=sPartName%></a>

<a href=javascript:window.open(" . /common/emxTree.jsp ?objectId
=<%= sPartId%>&mode=insert&jsTreeID=<%=jsTreeID%> ")
"><%=sPartName%></a>

```

Set the target frame to content, which is the parent frame for the treeDisplay frame. The first three examples above specify the content frame. Alternatively, you can open the tree in a new window, as shown in the last example. In this case, you do not specify a target frame.

Whenever the system calls a new tree, it automatically updates the right side frame, called detailsDisplay, with the URL and displays the new tree as the Categories menu. The URL is set in the href parameter of the menu object associated with the tree.

8. If you are working with the Web-based user interface as you are making changes and want to see your changes in the user interface, select  > Utilities > Reload Cache and click the browser Refresh button.

The cache refreshes automatically when the component age expires. This setting is in emxSystem.properties.

Only users assigned to the Administration Manager role have access to the Reload Cache tool.

Tree JavaScript APIs for Custom JSPs

This section lists JavaScript APIs you can use in custom JSP applications for working with nodes for a tree. The methods cover trees displayed in the Categories menu and trees displayed in the structure navigator.

In this section:

- [About the Tree JavaScript API for Custom JSPs](#)
- [Obtaining the Details Tree Object](#)
- [Obtaining the Structure Tree Object](#)
- [Getting the Selected Node](#)
- [Getting a Node](#)
- [Getting a Node's nodeID](#)
- [Getting a Node's Parent Node](#)
- [Getting Current Tree Root](#)
- [Getting Tree Current Root object ID](#)
- [Getting Tree Original Root object ID](#)
- [Setting the Selected Node](#)
- [Changing the Name of All Nodes Representing a Business Object](#)
- [Changing the Object ID of All Nodes Representing a Business Object](#)
- [Determining if an Object Exists in the Tree](#)
- [Deleting All Nodes Representing a Business Object](#)
- [Removing a Single Node](#)
- [Inserting a New Node into Details Tree](#)
- [Inserting a New Node into Structure Tree](#)
- [Deleting Nodes in Structure Trees](#)

About the Tree JavaScript API for Custom JSPs

To make sure all custom tree code is supported by future releases of Business Process Services, use only the listed APIs for any tree operation. Any API not included should be considered private and is not intended for use in custom JSPs.

If you are upgrading an application from a pre-version 10 software and have customized the code for navigation trees, you may have to migrate some code. For details, see "Migrating to Version 10 Tree API for Custom Pages" in the *Schema Reference Guide*.

Obtaining the Details Tree Object

The details tree (Categories menu) is always contained in the topmost frameset of the application.

To get a reference to the details tree object, use:

```
var objDetailsTree = top.objDetailsTree;
```

The following method is deprecated. If existing custom code uses it, the code will work for now, but the method will be eliminated eventually, so you should no longer use it.

```
var objDetailsTree = top.tempTree;
```

Obtaining the Structure Tree Object

The structure tree is always contained in the topmost frameset of the application.

To get a reference to the structure tree object, use the following:

```
var objStructureTree = top.objStructureTree;
```

Getting the Selected Node

You can retrieve the currently-selected node.

To retrieve the currently selected (highlighted) node in the structure navigator, use the following:

```
var objSelectedNode = [Tree].getSelectedNode();
```

Getting a Node

There are a variety of ways to get references to nodes, each returning either an emxUIObjectNode or emxUICategoryNode.

If you have the node's nodeID, you can use:

```
var objNode = [Tree].getNode(String nodeID);
```

To retrieve a node by its object ID, use this:

```
var objNode = [Tree].findNodeByObjectID(String objectID);
```

This method only returns the first node in the tree with this object ID.

To retrieve a node by its name, use this code:

```
var objNode = [Tree].findNodeByName(String name);
```

Getting a Node's nodeID

The nodeID is an internal unique identifier for a node.

To retrieve it, use:

```
var strNodeID = objNode.nodeID;
```

Getting a Node's Parent Node

You can obtain the parent node object of any given node using the method `getParent()`.

To obtain the parent node, using the `jsTreeID` (`nodeID`), use:

```
var parentNode = [Tree].getNode(nodeID).getParent();
```

To obtain the parent node for the current selected node, use:

```
var parentNode = [Tree].getSelectedNode().getParent();
```

Getting Current Tree Root

The current tree root is the root that is currently at the top of the tree, though not necessarily the original root of the tree.

To get the current root node, use:

```
var objRootNode = [Tree].getCurrentRoot()
```

Getting Tree Current Root object ID

You can retrieve the object ID of the current tree root.

To get the object ID of the current root, use:

```
var objRootNode = [Tree].getCurrentRoot()  
var strObjID = objRootNode.objectID;
```

Getting Tree Original Root object ID

You can retrieve the object ID of the original tree root.

To get the object ID of the original root, use:

```
var objRootNode = [Tree].getOriginalRoot()  
var strObjID = objRootNode.objectID;
```

Setting the Selected Node

You can set which node in the details tree should be selected.

To set the currently selected (highlighted) node in the details tree, use:

```
objDetailsTree.setSelectedNode(emxUIDetailsTreeNode node,  
    boolean refresh);
```

To set the currently selected (highlighted) node in the structure tree, use the following code:

```
objStructureTree.setSelectedNode(emxUIStructureTreeNode node,  
    boolean refresh);
```

To refresh the page after the setSelectedNode method call, use the following command:

```
parent.document.location.href=node.url;
```

Changing the Name of All Nodes Representing a Business Object

You can change the name of all nodes that represent a business object.

To change the name of all nodes representing a business object (for instance, all nodes that point to Part-5000), use:

```
[Tree].getObject(String objectID).changeName(String newName,  
    boolean refresh);
```

The following method is deprecated. If existing custom code uses it, the code will work for now, but the method will be eliminated eventually so you should no longer use it.

```
[Tree].getObject(String objectID).changeObjectName(String  
newName,  
    boolean refresh);
```

Changing the Object ID of All Nodes Representing a Business Object

You can change the object ID for all nodes that represent a business object.

To change the object ID of all nodes representing a business object (for instance, all nodes that point to Part-5000), use:

```
[Tree].getObject(String objectID).changeID(String newObjectID,  
    boolean refresh);
```

The following method is deprecated. If existing custom code uses it, the code will work for now, but the method will be eliminated eventually so you should no longer use it.

```
objNode.changeObjectID(String newObjectID, boolean refresh);
```

Determining if an Object Exists in the Tree

You can determine if an object exists within a tree.

To determine if a specific object exists in the structure tree, use:

```
boolean result = [Tree].hasObject(String objectID);
```

Deleting All Nodes Representing a Business Object

You can delete all nodes for a business object from trees.

To delete all nodes representing a business object (for instance, all nodes that point to Part-5000), use:

```
[Tree].deleteObject(String objectID, boolean refresh);
```

Removing a Single Node

There are two ways to remove a single node: using its `nodeID` or its `objectID`.

To delete using the `nodeID`, use:

```
objParentNode.removeChildByID(String nodeID, boolean refresh);
```

To remove a node by using the object ID, use:

```
objParentNode.removeChild(String objectID, boolean refresh);
```

Inserting a New Node into Details Tree

You can insert a new node into a tree.

To insert a new node (after creating/connecting a new object) in the details tree, use the following methods:

```
top.addDetailsTreeNode(String objectId, String jsTreeID);
top.addDetailsTreeNode(String objectId, String jsTreeID,
                      String emxSuiteDirectory);
top.addDetailsTreeNode(String objectId, String jsTreeID,
                      String urlParams);
```

This method adds the new node along with the categories defined in the administrative menu object. The menu used for building the new tree is based on the objectId.

emxSuiteDirectory is an optional parameter to insert an application-specific alternate tree.

urlParams is an optional parameter to add any additional URL parameters like "treeLabel=TaskNote&emxSuiteDirectory=engineeringcentral".

The following method is deprecated. If existing custom code uses it, the code will work for now, but the method will be eliminated so you should no longer use it.

```
top.addContextTreeNode(String objectId, String jsTreeID);
top.addContextTreeNode(String objectId, String jsTreeID,
                      String emxSuiteDirectory);
```

Inserting a New Node into Structure Tree

You can insert a new node into a structure tree.

To insert a new node (after creating/connecting a new object) in the structure tree, use the following methods:

```
top.addStructureTreeNode(String objectId, String  
parentObjectId,  
    String jsTreeID);  
top.addStructureTreeNode(String objectId, String  
parentObjectId,  
    String jsTreeID, String emxSuiteDirectory);  
top.addStructureTreeNode(String objectId, String  
parentObjectId,  
    String jsTreeID, String urlParams);
```

`emxSuiteDirectory` is an optional parameter to insert an application-specific alternate tree.

`urlParams` is an optional parameter to add any additional URL parameters like
`"treeLabel=TaskNote&emxSuiteDirectory=engineeringcentral"`.

The following method is deprecated. If existing custom code uses it, the code will work for now, but the method will be eliminated so you should no longer use it.

```
top.addStructureNode(String objectId, String parentObjectId,  
    String jsTreeID);  
top.addStructureNode(String objectId, String parentObjectId,  
    String jsTreeID, String emxSuiteDirectory);
```

Deleting Nodes in Structure Trees

You can delete nodes from the structure tree.

To delete all nodes representing a business object (for instance, all nodes that point to Part-5000) in both the structure tree, use:

```
top.deleteObjectFromTrees(String objectId, Boolean refresh);
```

The following method is deprecated. If existing custom code uses it, the code will work for now, but the method will be eliminated eventually so you should no longer use it.

```
top.deleteStructureNode(String objectId, Boolean refresh);
```

Deprecated Methods (Navigation Tree)

The following table lists deprecated methods for navigation trees. These methods will work for now but you should not use them for any new implementations.

Deprecated Method Name	Replaced by New Method
var objDetailsTree = top.tempTree;	var objDetailsTree = top.objDetailsTree;
changeObjectName(String newName, boolean refresh);	changeName(String newName, boolean refresh);
changeObjectID(String newObjectID, boolean refresh);	changeID(String newObjectID, boolean refresh);
top.addContextTreeNode(String objectId, String jsTreeID); top.addContextTreeNode(String objectId, String jsTreeID, String emxSuiteDirectory);	top.addDetailsTreeNode(String objectId, String jsTreeID); top.addDetailsTreeNode(String objectId, String jsTreeID, String emxSuiteDirectory); top.addDetailsTreeNode(String objectId, String jsTreeID, String urlParams);
top.addStructureNode(String objectId, String parentObjectId, String jsTreeID); top.addStructureNode(String objectId, String parentObjectId, String jsTreeID, String emxSuiteDirectory);	top.addStructureTreeNode(String objectId, String parentObjectId, String jsTreeID); top.addStructureTreeNode(String objectId, String parentObjectId, String jsTreeID, String emxSuiteDirectory);
top.deleteStructureNode(String objectId, Boolean refresh);	top.deleteObjectFromTrees(String objectId, Boolean refresh);

Parameters for Tree Menu Objects

This table describes the available parameters and settings for menu objects that represent navigation trees for object types.

For specific instructions on how to create menu objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Alt	Not applicable to menu objects for trees because the root node for a tree doesn't have a ToolTip.	--
Commands (specified in the Items tab in Business Modeler)	The command objects that represent the categories in the tree. The order of the commands is the order the categories appear in the trees in the user interface.	Names of command objects, such as: ENCBOMTreeCategory AEFRelToItemTreeCategory TMCRoutesTreeCategory ENCPartDesignInfoTreeCategory
href	The URL executed when a user clicks the root node for the tree. It is the Properties page for the object by default. This page is always displayed in the right frame, called "detailsDisplay", and not in another window or frame. The value for the href parameter should be a JSP and any associated parameters. You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components .	emxDynamicAttributes.jsp \${COMMON_DIR}/emxPartInfo.jsp \${SUITE_DIR}/emxViewPartAttr.jsp
Icon	Icon for the object within ENOVIA Live Collaboration. This setting is NOT the image that displays next to the root node on the dynamic user interface. The emxFramework.smallIcon.SYMBOLIC_NAME property in emxSystem.properties is used to specify that image. For more information, see About the Icon for a Type's Tree .	The name of an image file, such as Part.gif.
Label	The text to label the tree root node. The label can contain: <ul style="list-style-type: none"> Any of the regular expression macros supported by applications are evaluated at run time to display the current object type name and revision. Static text. String resource ID defined in the string resource property file. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components. The system initially considers every label text as a string resource ID. If it finds the value for this ID in the resource file, it displays the obtained value. If not, the text is displayed as is. There are two other ways to define the label for a tree node: using the TreeLabel URL parameter passed to emxTree.jsp and using a JPO (specified with the Label Program and Label Function settings). The order of precedence in which the tree handles these methods of getting the root node label is: <ul style="list-style-type: none"> TreeLabel URL parameter--Overrides all other methods. JPO--Used if TreeLabel parameter is not defined. Label parameter--Used only if TreeLabel and JPO are not defined. 	\$<type> \$<name> \$<attribute[attribute_Weight].value> Engine - \$<type> \$<name> Connected ECR emxFramework.Common.Part
Menus	Not applicable to menu objects for trees because trees do not have submenus.	--
Settings	Additional settings that define the behavior and appearance of the tree.	Name/value pairs, as defined in Settings for Tree Menu Objects .

Settings for Tree Menu Objects

This table lists and describes the settings for menus that represent navigation trees. Note that the names and values are case sensitive.

Setting	Description	Accepted Values/Examples
Currency Converter	Specifies whether the target page should include the Currency Converter tool. In the href URL called when the user clicks the tree root node, the system passes a parameter called CurrencyConverter and includes the value specified for this setting. If the value is True, the target page includes the Currency Converter tool. If this setting is not included, the value is assumed to be false and the target page does not display the Currency Converter tool.	True False
Default Category	<p>Specifies the category to be selected when an object is selected for display, or when the user clicks the top node in the Categories menu. By default, the root node is selected, which means the object's Properties page displays. If the tree contains a category that users frequently want to see, specify it instead.</p> <p>Alternatively, you can pass the DefaultCategory parameter to emxTree.jsp. If both are defined, this URL parameter overrides the setting.</p> <p>Note that a tree does not look for default categories defined for its sub-trees (assigned sub-menus).</p>	<p>The name of a category command object to select when the tree first opens. The command object must be assigned to the tree menu object and if it is not, the root node is selected.</p> <p>Default Category=PMCWBS</p>
Dynamic Command Function	Defines the method in the JPO specified by the Dynamic Command Program setting that returns a list containing the categories to include in the navigation tree.	<JPO Method Name>
Dynamic Command Program	Defines the JPO invoked from a navigation tree component. This setting can only be used on static menus or commands in a tree; it cannot be used with commands that have been dynamically added.	<JPO Name>
Help Marker	<p>Specifies the name of the help marker to call for context-sensitive help. For information about implementing help, see About Context-Sensitive Help.</p> <p>In the href URL called when the tree root node is clicked, the system passes a parameter called HelpMarker and includes the marker text specified for this setting.</p>	The naming convention for help markers "emxpath" followed by the object or feature and then the action, for example, emxhelprouutecreate and emxhelpprojectedit. The marker is all lowercase with no spaces.
Label Function	Specifies the function in the JPO specified in the Label Program setting that gets the value for the root node label.	The name of a method in the Label Program JPO.
Label Program	<p>Use to define the label for the root node using a JPO. This parameter specifies a JPO that contains a method to get the value for the label. The method is specified using the Label Function setting.</p> <p>The input for this JPO must include:</p> <ul style="list-style-type: none"> -A list of all of the Request Parameters in a HashMap -Method Name as a string -JPO Program Name as a string <p>The output should be the string that is returned to the tree and used for the label.</p> <p>There are two other ways to define the label for a tree node: using the TreeLabel URL parameter passed to emxTree.jsp and using the Label parameter for the menu object. For information on the precedence order, see the Label parameter.</p>	The name of a JPO added to the database. If a JPO is configured for a tree category or root node label and the JPO returns an empty string or null, the system throws an exception.
Message URL	Specifies the label to use for URL links that call the tree.	The value can be any select statement,

Label	<p>For example, notifications that users have new tasks include URL links that open the Inbox Task tree for the user's task. By default, the URL link label shows the name of the Inbox Task object, which is an autogenerated name. But if the route creator entered a name for the task, it's better to show that entered name for the URL link.</p> <p>To have the URL link show the task's entered name, you would add the Message URL Label setting to the tree menu object for the Inbox Task. The value would be a select macro for the attribute that contains the entered name, in this case the Title attribute.</p> <p>When building the URL link, the system first looks for the Message URL Label setting on the application specific tree menu, for example, ENctype_InboxTask. If not found there, the system looks for the setting in the common tree for that object type. If the common tree doesn't contain the setting, the system defaults to the object's Type,Name,Revision (for example, Inbox Task IT-0000101).</p>	<p>plain text or a string resource ID. For example:</p> <pre>\$<attribute[attribute_Title].value></pre> <p>Task Title</p> <pre>emxEngineeringCentral.TaskTitle</pre> <p>The string resource Id can contain macros also, as in the following example:</p> <pre>emxEngineeringCentral.TaskTitle=\$<type> \$<name> \$<revision>: Urgent Task</pre>
Printer Friendly	<p>Passes the PrinterFriendly parameter and the entered value to the JSP specified in the href URL. JSPs that use the PrinterFriendly parameter, such as emxTable.jsp and emxForm.jsp, show the Printer Friendly tool when the setting is True and hide it when False. If the setting is not included, emxTable.jsp and emxForm.jsp show the tool by default. Users can click the tool to get a version of the current page that can be printed with the browser's Print button.</p> <p>Note that you can also specify the PrinterFriendly parameter in the href URL for JSPs that use it.</p>	<p>True (default) False</p>
*Registered Suite	<p>The application the menu belongs to. The system looks for files related to the tree in the registered directory for that application, which is specified in emxSystem.properties.</p> <p>Based on the application name, the system passes the following parameters in the href URL:</p> <ul style="list-style-type: none"> suiteKey emxSuiteDirectory StringResourceFileId 	<p>Set the value with no spaces, for example, EngineeringCentral or Framework. Set the value to the suite name as defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. If the suite name starts with "eServiceSuite", you can skip this prefix and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite".</p> <p>In the href URL called when the user clicks the tree root node, the system passes a parameter called "suiteKey". The value for the parameter is the property name from emxSystem.properties that maps to the setting's value.</p>
Structure Menu	Specifies the menu administrative object that defines a structure tree for the object type.	The name of a menu administrative object.
Tip Page	Specifies whether the target page should include the Tip Page tool and the URL to call when the user clicks the tool. In the href URL called when the user clicks the tree root node, the system passes a parameter called TipPage and includes the URL for this setting. If this setting is not included, the target page does not display the Tip Page tool.	Name of a custom html or JSP page, including any path. The starting point for the directory reference is the content directory. For example, if you want to call an html file in ematrix/doc/customcentral and the content directory is ematrix/customcentral, add this parameter to the table.jsp: TipPage=../doc/customcentral/tippage.html
Tree Scope ID	<p>Passes the current tree's objectId to all its categories, to subtrees inserted into the tree, and to categories in the subtrees, to the nth level, in the parameter name specified as the value for this setting. This setting is valid only for tree menu objects and not for tree category command objects.</p> <p>For example, suppose you want to pass the object ID for</p>	<p>The name of the parameter you want to use to pass the object ID. For example:</p> <ul style="list-style-type: none"> workspaceId projectId partId

	<p>the current workspace to all categories and subtrees in the workspace tree. You want to pass the object ID using a parameter called workspaceId. Add the Tree Scope ID setting to the menu administrative object for workspace trees (which is called type_Project due to administrative object renaming) and enter "workspaceId" as the value. The parameter and value "workspace Id=OBJECTID OF THE WORKSPACE will be available to all the categories and subtrees of the current workspace to the nth level until another workspace object gets inserted or until any explicit URL parameter with the same name "workspaceId" is passed to the tree.</p>	<p>You can also pass specific parameters from the tree menu object to its categories using the AppendParameter parameter for the emxTree.jsp.</p>
Type Icon Function	Specifies the function in the JPO specified in the Type Icon Program setting that retrieves an icon to show in addition to the icon defined by the emxFramework.smallIcon.type system property.	Function Name
Type Icon Program	Defines the program that contains the function specified using the Type Icon Function setting.	JPO Name

*Required Setting

Parameters for Tree Category Command Objects

This table describes the available parameters for command objects used for tree categories. These apply to command objects for standard navigation trees (the *Categories* menu) and for structure trees.

For specific instructions on how to create command objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Icon	Icon for the object within ENOVIA Live Collaboration. This setting does not display an image in the UI.	The name of an image file, such as Part.gif.
Label	<p>The text that should be used to represent the tree category. The label can be made up of:</p> <ul style="list-style-type: none"> Plain text, such as "ECRs". Text and/or any valid select expression that gets evaluated before displaying the label, such as "Subfolder (\$<attribute[attribute_Count].value>)". <p>For example, you can configure the label to show a count of the number of items in the category to the right of the label and in parentheses. If the category is subfolders of a folder and a particular folder has three subfolders, the label would show "Subfolders (3)".</p> <p>To configure the label to include a count, use the macro \$<attribute[attribute_Count].value>. For example:</p> <pre>Label = Subfolder (\$<attribute[attribute_Count].value>)</pre> <p>The Count attribute is assigned to the Workspace Vault type (called folder in the UI). This attribute keeps the count of the number of subfolders created below the folder. At run time, this value is evaluated and is shown in the label for the tree category.</p> <ul style="list-style-type: none"> A string resource ID, such as "emxEngineeringCentral.DesignTOP.CreatePart". <p>To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components. The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it does not find one, it displays the entered text and/or output from the select expression.</p> <p>To internationalize a label that needs a select expression, assign a string resource ID to the label. The string resource ID can contain the macro to be evaluated. For example, using the use case described in option 2 above, you could configure the command object label as follows:</p> <pre>Label = emxFramework.Common.Subfolder</pre> <p>And in string resource file:</p> <pre>emxFramework.Common.Subfolder = Subfolder (\$<attribute[attribute_Count].value>)</pre> <p>Tree category labels can also be defined using a JPO. See the Label Function and Label Program settings. If these JPO settings are defined for a category, they override any label defined using this parameter.</p>	ECRs Subfolder (\$<attribute[attribute_Count].value>) emxEngineeringCentral.DesignTOP.CreatePart
href	<p>The URL that gets executed when the user clicks the tree category. This URL displays in the right frame and not in another window or frame.</p> <p>The value for the href parameter should be a JSP and any associated parameters. You can specify the path of the JSP using any of the standard directory macros or you can</p>	emxECRs.jsp \${COMMON_DIR}/emxECRs.jsp \${SUITE_DIR}/emxECRs.jsp \${ROOT_DIR}/emxECRs.jsp

	leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components .	
Alt	Not applicable for tree categories because ToolTips do not display for categories.	--
Access	The persons, roles, and groups who can access the tree category. To make the tree category available to all users, regardless of role/group assignments, choose All. Note that if no users are assigned access, the system assumes all users have access.	Names of group, role, person administrative objects. or All (default)
Settings	Additional settings that define the behavior and appearance of the command. For a list of the accepted settings, see the table below.	Name/value pairs, as defined in Settings for Tree Category Command Objects .

Settings for Tree Category Command Objects

This table lists and describes the settings for command objects used for navigation tree categories. Note that the names and values are case sensitive.

Setting	Description	Accepted Values/Examples
*Registered Suite	<p>The application the command belongs to. The system looks for files related to the tree category in the registered directory for that application, which is specified in emxSystem.properties.</p> <p>Based on the application name, the system passes the following parameters in the href URL:</p> <ul style="list-style-type: none"> suiteKey emxSuiteDirectory StringResourceFileId 	<p>Set the value without any spaces, for example, EngineeringCentral or Framework. Set the value to the suite name as defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. If the suite name starts with "eServiceSuite", you can skip this prefix and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite".</p> <p>In the href URL called when the user clicks the tree category, the system passes a parameter called "suiteKey". The value for the parameter is the property name from emxSystem.properties that maps to the setting's value.</p>
Label	<p>The label that shows the list of structures in the Structure Selector combo box.</p> <p>If there is only one command attached to the structure menu the system ignores this setting.</p>	<p>A string resource ID or text string: emxTeam.common.Folders Discussions EBOM</p>
PreExpand	<p>Determines whether to let the JPO program decide whether to add the [+] icon or to simply add the [+] without running the JPO.</p>	<p>true--The system runs the Structure Program and Function defined for the Structure command to retrieve the list of objects. It runs the same program for each object to determine whether it must display a [+] icon. If the list of objects is greater than 1, it adds a plus [+] to the structure node.</p> <p>false (default)--The system runs the JPO Program and Function one time to get the list of objects. It displays plus icon [+] for each structure node.</p>
Revision Filter	<p>Enables or disables the display of the revision filter button.</p>	<p>Enable Disable (default)</p>
Structure Function	<p>JPO method name that gets the object list.</p>	<p>The name of a method in the JPO specified in the Structure Program setting, such as: getWorkspaceFolders getEBOMs</p>
Structure Inquiry	<p>Inquiry administrative object name that gets the object list to display within the structure. This is an alternative to the JPO approach and the JPO approach takes precedence.</p> <p>This setting is not implemented.</p>	<p>The name of an inquiry administrative object: SCSWspFolderStructure ENCEBOMList</p>
Structure Program	<p>JPO program that contains the method that retrieves the object list to display in the structure. The method is specified in the Structure Function setting.</p> <p>For instructions on writing the JPO program, see Guidelines for Writing Structure Tree JPO.</p>	<p>The name of a JPO program added to the database. SCSWorkspaceStructure ENCBOMStructure</p>
Target	<p>Controls where the page specified in the href parameter</p>	<p>content--The page replaces the content</p>

Location	<p>appears.</p> <p>Popup is the default value for a textbox and is used if the setting is not included or if the named frame cannot be found.</p> <p>This value can be set to any valid frame name that is available to the form body frame.</p> <p>When using slidein as the Target Location and the Popup Modal setting is true, then the content window and the global toolbar are grayed out and disabled until the slidein window is closed.</p>	<p>frame.</p> <p>popup--Page appears in new window. Set the modality of the window using the Popup Modal setting.</p> <p>_top--Page replaces the entire body of the browser window.</p> <p>listHidden--Carries out any background processing when no UI is required.</p> <p>hiddenFrame--Targets the Navigator hidden frame to perform any background processing. Do not use this frame within the context of the configurable table. Use listHidden instead.</p> <p>mainFrame--Page appears in the frame that includes the content and menu frames.</p> <p>fromViewHidden--Used within the context of the configurable form.</p> <p>slidein--Page appears in a slidein frame (slides in along the right side of the window).</p> <p>This value can be set to any valid frame name that is available to the form body frame.</p>
----------	---	--

*Required Setting

URL Parameters Accepted by emxTree.jsp

This table lists the parameters you can specify for emxTree.jsp. When you call emxTree.jsp from a JSP, you can add these parameters. Note that the name of the menu object for the tree is not required because the system automatically looks for the tree menu object that has the same name as the symbolic name of the object's type.

Parameter	Description	Accepted Input Values
AppendParameters	<p>Passes common parameters to the tree categories in the tree. To use the parameter, set it to true and then include the parameters to pass. Parameters added to emxTree.jsp that are not with AppendParameters are passed only to the root tree node and not to categories.</p> <p>You can also use the Tree Scope ID setting for the tree menu object to pass the current tree's object ID to all categories and subtrees.</p>	<p>To make Param1 and Param2 available to the children categories:</p> <pre><a href="../common/emxTree.jsp?objectId=<%=sPartId%>&mode=insert&jsTreeID=<%=jsTreeID %>&AppendParameters=true&Param1=Value1&Param2=Value2" class="object" target="content"><%=sPartName%></pre> <p>To make ProjectId and Workspace available to the tree categories:</p> <pre><a href="../common/emxTree.jsp?objectId=<%=sPartId%>&mode=insert&jsTreeID=<%=jsTreeID %>&AppendParameters=true&ProjectId=<%=sProjId%>&Workspace=<%=sWorkspace%>" class="object" target="content"><%=sPartName%></pre> <p>To make ProjectId and Workspace available only to the root tree node and not to the tree categories:</p> <pre><a href="../common/emxTree.jsp?objectId=<%=sPartId%>&mode=insert&jsTreeID=<%=jsTreeID %>&&ProjectId=<%=sProjId%>&Workspace=<%=sWorkspace%>" class="object" target="content"><%=sPartName%></pre>
DefaultCategory	<p>Specifies the category to be selected when the tree first opens. The root node is selected, by default, so the object's Properties page displays. If the tree contains a category that users frequently want to see, use this parameter to select it instead.</p> <p>Alternatively, you can use the Default Category setting for the tree's menu object. If both are defined, this URL parameter overrides the setting.</p> <p>Note that a tree does not look for default categories defined for its subtrees (assigned sub-menus).</p>	<p>The name of a category command object to select when the tree first displays. You must assign the command object to the tree menu object. If it is not, the root node is selected.</p> <p>DefaultCategory=PMCWBS</p>
jsTreeID	<p>The ID of the existing tree object required to update the tree with "insert" mode.</p> <p>When the system</p>	node567590204694.5947

	<p>constructs the tree for the first time, the "jsTreeID" parameter does not have any valid value.</p> <p>Whenever the user clicks a category within the tree, the system passes the current "jsTreeID" to that page. If that page needs to update the tree, it must call emxTree.jsp with the same "jsTreeID" value.</p>	
mode	Controls whether the tree for the object is inserted into the current tree or replaces the current tree.	Insert--Insert the tree for the selected object into the current tree, if there is one. Replace (default)--Replace the current tree, if there is one, with the tree for the selected object.
objectId	This is the business object ID that the system will construct the tree for. For example, if the object is of type ECR, the system will construct the tree object of menu type "type_ECR". If the menu "type_ECR" does not exist, the system will base the tree on the "Default Tree" configuration.	2233.5567.2323.4678
targetLocation	When a URL is defined to open in the slide-in frame (the Target Location setting for the component is set to slidein), BPS appends this URL parameter with this value. This parameter supports a custom page where the html for a popup window needs to be different than the html for a slide-in window.	slidein
treeLabel	<p>Use to override the default label for a tree (defined in the "Label" parameter of the tree menu administrative object). When this parameter is available to the emxTree.jsp page, the page uses this value as the Label for tree menu base node.</p> <p>This parameter also overrides any JPO specified for the label in the settings</p>	emxTree.jsp?treeLabel=custom Part Label&objectId=3434.345.4564.7755

	<p>Label Program and Label Function.</p>	
treeMenu	<p>Use to override the standard tree menu for the object's type (or any alternate tree defined by an application). The standard tree menu for an object is the symbolic name of the type. For example, the standard tree menu object for quality part plans is <code>type_QualityPartPlan</code>. For more information on calling specific trees, see Specific Trees for an Object Type.</p> <p>Using a custom tree menu by overriding the system-defined tree for any object type is NOT RECOMMENDED.</p>	<p>Name of a menu object for a tree: <code>customPartTree</code></p>

About the Default Tree

Business Process Services installs with a default tree that contains general categories used by many types. The system uses the default tree for any type that does not have a tree defined for it (or defined for any type higher in the hierarchy).

For details on how the system determines which tree menu object to use for an object, see [Specific Trees for an Object Type](#).

You specify the tree to use as the default tree using the following property in emxSystem.properties. The value for the property should be the symbolic name of the menu object that you want to use as the default tree.

```
#Setting to pick up default tree name  
emxNavigator.DefaultTree.TreeName = menu_DefaultTree
```

You can configure the default tree as you would any tree: change the categories by changing the connected command objects, changing the image for it, and changing parameters and settings for the root node and command objects. For more information, see [Building a Navigation Tree](#).

Specific Trees for an Object Type

You can use the background information in this section to define which tree should be called for a specific object type.

Typically, when a user clicks an object's name in the ENOVIA products to see details about the object, emxTree.jsp uses the tree menu object that has the same symbolic name as the object's type. For example, when a user clicks the name of a Part Quality Plan, the system uses the tree menu object named type_PartQualityPlan. The menu object that matches the symbolic name for a type is referred to as the standard tree menu for an object.

There are cases when the standard tree menu for an object is not suitable. For example, some object types are used by more than one ENOVIA product and each application may have a unique tree menu. In this case, the application that owns that object type installs the standard tree menu object and all other applications that use the type install an alternate tree menu for the type. The owner application is the one that has primary responsibility for creating and managing that type of object. For example, most ENOVIA products use parts but Engineering Central is the owner of the Part type. Therefore, Engineering Central installs the type_Part menu and other applications install an alternate menu for parts. Specific installations may also need to define custom tree menus. (Using custom tree menus instead of the system's standard or pre-defined alternate menu is not recommended.)

When a user clicks a link to view details for an object, the system uses the following methods to determine which tree to display for the object, in this order of precedence:

1. **Override all trees using treeMenu parameter**--The system first looks for the treeMenu parameter passed to emxTree.jsp URL. The value for the treeMenu parameter is the name of a menu object for a navigation tree and you can use the parameter to override all other trees that could be used for the object.

Use the treeMenu parameter when you want to use a tree menu other than the standard (as described in item 3 below) or alternate (as described in item 2 below) tree menu for the object's type. Also use the treeMenu parameter to call an application's alternate tree menu for an object when the link is external to the application. For example, if the link is within IconMail or email, the system will not pick up an application's alternate menu because the link is not within a particular application (so it would not know which application's alternate tree menu to use).

2. **Use application's alternate tree**--If no treeMenu parameter is passed, the system gets the suite directory that called emxTree.jsp and checks if that application has an alternate tree menu defined for the object type. ENOVIA products define alternate trees for objects used by more than one application. The alternate tree menus need not be assigned to the top-level menu called Tree but the standard tree menus must be assigned to the top-level menu.

Application-specific alternate menus are defined using the property SUITEKEY.emxTreeAlternateMenuItemName.TYPE_SYMBOLIC_NAME in the application's property file, such as emxEngineeringCentral.properties. SUITEKEY is the same as the value assigned to the property eServiceSuites.DisplayedSuites in the emxSystem.properties file. The naming convention for the alternate menus is the type's symbolic name prepended with the 3-letter abbreviation for the application, for example, ENCtype_Part. If the specified alternate menu does not exist in the database, the standard tree menu for that type, for example type_Part, is used. (The naming conventions for the application abbreviations are listed in [Naming Conventions for UI Administrative Objects](#).)

For example, the emxEngineeringCentral.properties file contains this property:

```
eServiceSuiteEngineeringCentral.emxTreeAlternateMenuItemName.type_P  
art=ENCtype_part
```

In special cases, such as when forwarding or redirecting the response to emxTree.jsp, the alternate tree menu may not display correctly. In these cases, you must pass the URL parameter "emxSuiteDirectory" with the appropriate directory to emxTree.jsp. This directory is the same as defined in the emxSystem.properties (SUITE_NAME.Directory =), where the value is the application directory installed below the ematrix directory.

3. **Use type's standard tree**--If you have not defined an application-specific alternate menu for the object type, the system looks for a menu object with the same name as the type's symbolic name. If it does not find one, the system looks for a menu object with the symbolic name for the type's parent type, then grandparent type, and so on until it reaches the top of the type hierarchy.
4. **Use Default Tree**--If there is no menu object for the object's type or parent types, the system uses the Default Tree menu object. For more information on the default tree, see [About the Default Tree](#).

About the Icon for a Type's Tree

When displaying the tree for an object type, the system uses the image file specified in the emxSystem.properties file for that object type.

For example, this property defines the icon for a mechanical part:

```
emxFramework.smallIcon.type_MechanicalPart = iconSmallMechanicalPart.gif
```

This property is also used to display an icon for the type (in addition to the column's data) in a table column when the Show Type Icon setting is true for the column. If there is no property defined for the type's icon, the system looks for a property for the type's parent type icon, then grandparent's type and so on. If no property is defined for any parent type's icon, the system uses the default image, defined by this property:

```
# default icon name for the type  
emxFramework.smallIcon.defaultType = iconSmallDefault.gif
```

One exception to the above is that if the system uses a parent's tree menu because no tree menu is defined for the subtype, then it uses the icon associated with the parent type instead of the subtype.

In addition to using the icon as defined above, you can define an additional icon based on a business rule to show in the root node of the tree. For example, you could show different icons depending on the object's lifecycle state. The emxSystem.properties file lets you define a JPO:method to execute the logic and return the required icon:

```
emxFramework.UICOMPONENTS.OverrideTypeIcon.Program=<JPO Name>:<Method Name>
```

This property defines a global JPO:method to use for fields and columns (that define the **Show Type Icon** or **Show Alternate Icon** settings) and all navigation trees. To use a different JPO:method for a specific tree, use the **Type Icon Function** and **Type Icon Program** settings when defining the tree (and an appropriately coded JPO program:method).

The function that retrieves the custom icons to display for the root node in the tree must return an image name or path for the id that is passed to it. The path, including the image name, must be specified relative to the ematrix\common directory. The image must be a gif image, and cannot include any spaces or special characters.

You cannot override the Type icon for the generic create form component, emxCreate.jsp.

For details about the JPO method signature, input arguments, returned values and sample JPO code, see [Column Values Including Additional Icons](#) in the Tables section.

Dynamic Expand for Trees

You can configure a tree category so a list of business objects is inserted under it automatically, without users having to first view details for objects.

In this section:

- [About Dynamic Expand for Tree Categories](#)
- [Requirements for a Dynamic Expand JPO](#)
- [Examples of Dynamic Expand Inquiries](#)

About Dynamic Expand for Tree Categories

You can configure a tree category so a list of business objects is inserted under it automatically. For example, Folder categories are often configured to display all folders without users having to view the folders first.

This feature is recommended primarily for hierarchically organized categories such as Folders and Subfolders. For best performance and consistency of behavior across applications, the preferred method of navigation is to use the standard tree behavior that does not use the Expand Inquiry. If you require this setting, restrict its use to categories containing relatively few objects.

There are two ways to get the list of business objects to insert under the category:

- JPO and method

The system obtains the objects using a method in a JPO. You specify the JPO using the Expand Program setting, and you specify the method using the Expand Function setting for the command object. If you add both the JPO and inquiry settings for a category command object, the JPO takes precedence.

- Inquiry administrative object

The system obtains the objects from the database using an inquiry object. You configure this behavior using a setting called Expand Inquiry for the command object that represents the category. The value for the setting is the inquiry object name. For example:

Expand Inquiry=TMCFolders

Requirements for a Dynamic Expand JPO

To specify a JPO and method that gets the list of business objects to insert under a dynamic expand tree category, use the Expand Program and Expand Function settings for the command.

The input for the JPO must include:

- A list of all of the Request Parameters in a HashMap
- Method Name as a string
- JPO Program Name as a string

The output should be a maplist that contains the list of business object IDs to process and insert under the category. The list of IDs determines which types of object are being processed and which tree to use for the objects.

Examples of Dynamic Expand Inquiries

You can create the required inquiry administrative object using MQL scripts or Business Modeler. Examples for creating the inquiry by MQL are shown below.

For a description of the parameters available for inquiry objects, see [Parameters for Table Objects](#).

The following examples are provided:

- [Inquiry Example: TMCFolders](#)
- [Inquiry Example: ENCPartList](#)
- [Inquiry Example: ENCBOMList](#)

Inquiry Example: TMCFolders

```
*****  
# Create the inquiry object for Folder List.  
add inquiry TMCFolders  
code "expand bus \"${ID}\" from relationship \"${WS_VAULT}\"  
select bus id dump |  
where '( (owner == \"${USER}\") || (current.access[read] ==  
true))'  
    argument WS_VAULT relationship_ProjectVaults  
    pattern *|*|*|*|*|${ OID }  
    format ${OID}  
*****
```



Inquiry Example: ENCPartList

```
*****  
# Create the inquiry object for Part List Table.  
add inquiry ENCPartList  
code "temp query bus ${PART} * * select id dump |"  
    argument PART type_Part  
    pattern *|*|*|${OID}  
    format ${OID}  
;  
*****
```



Inquiry Example: ENCBOMList

This example, ENCBOMList, uses \${ID} as a macro but this is not same as the Runtime Program Environment (RPE) variable \${OID}. This \${ID} macro gets substituted at run time, if there is a valid object Id available to emxTree.jsp.

```
*****  
# Create the inquiry object for BOM Table for a specific Part  
add inquiry ENCBOMList  
code "expand bus '${ID}' from relationship '${EBOM_REL}' terse  
select relationship id dump |"  
    argument EBOM_REL relationship_EBOM  
    argument ID dummy  
    pattern *|*|*|*|*|${BUSID}|${RELID}  
    format ${RELID}~${BUSID}  
;  
*****
```

Guidelines for Writing Structure Tree JPO

If you define a JPO for a structure tree category, it must contain the business logic to fetch the object list required for structure navigation.

The JPO class must have the mxMain method implemented. The JPO methods are defined with the following signature.

```
public static Object methodName(matrix.db.Context context,  
String[] args)  
throws Exception  
{  
}
```

The input arguments are:

- Context--passed in always.
- String[] args--will have one element of type HashMap.

The input argument "arg" is a HashMap, which contains the details of the object to be processed. The data structure of this input parameter (HashMap) is defined below:

Key Name	Data Type	This key is assigned to the:
languageStr	String	Language string for the current browser setting. The program can use this key for any internationalization purpose.
New Value	String	Applicable only for use with Update Program and Update Function. Key "New Value" is assigned to the new value changed by the user.
objectId	String	Business object Id (OID) to be used in the current form page.
Old Value	String	Applicable only for use with Update Program and Update Function. Key "Old Value" is assigned to the actual value for the field before the user changed it.
relId	String	Relationship id (RelID) to be used for the form page.
requestMap	HashMap	Key "requestMap" is assigned to a HashMap, which contains a set of key/value pairs as available in the request object. The keys are of type String (parameter names) and the values are of type String (parameter value).

The JPO method can extract the information from the input argument for processing the data.

```
public static Object methodName(matrix.db.Context context,  
String[] args) throws Exception  
{  
HashMap programMap = (HashMap) JPO.unpackArgs(args);  
HashMap requestMap = (HashMap) programMap.get("requestMap");  
HashMap paramMap = (HashMap) programMap.get("paramMap");  
String objectId = (String) requestMap.get("objectId");  
String relId = (String) paramMap.get("relId");  
String languageStr = (String) requestMap.get("languageStr");  
?  
return object  
}
```

The method returns an object of appropriate type depending on the where it is used. The details of return types are explained in the following sections.

Accessing Applications Externally

You can configure links on external Web applications and email messages that let people access applications based on the dynamic user interface.

In this section:

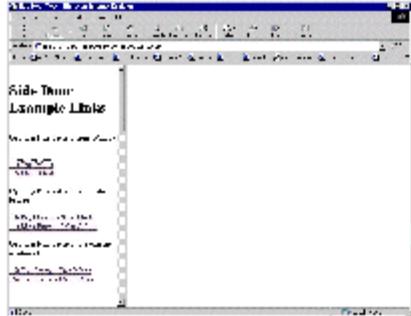
- [About External Access to Applications](#)
- [Content of the Navigator Page](#)
- [Location of the Navigator Page](#)

About External Access to Applications

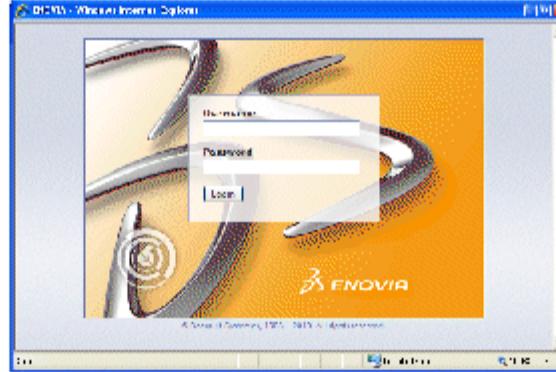
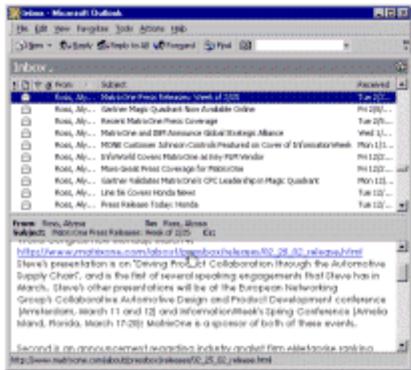
You can configure links on external Web applications and email messages that let people access applications based on the dynamic user interface. The link must call the main Navigator page for the dynamic user interface, emxNavigator.jsp.

When a user clicks the link and is already logged in (and there is still a valid session context), the system launches the Navigator page directly. If the person is not logged in, the system presents the Login page. After submitting a successful login, the Navigator page displays.

Link to Dynamic UI application on external Web site

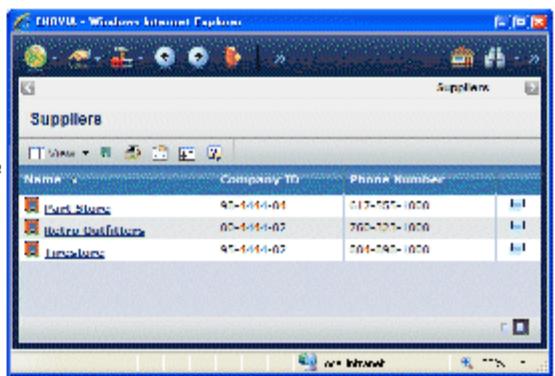


Link to Dynamic UI application in email



Goes to Login page if user is not logged in

Goes to Navigator page



This section describes how to configure the link so the Navigator page displays the way you want it to.

Content of the Navigator Page

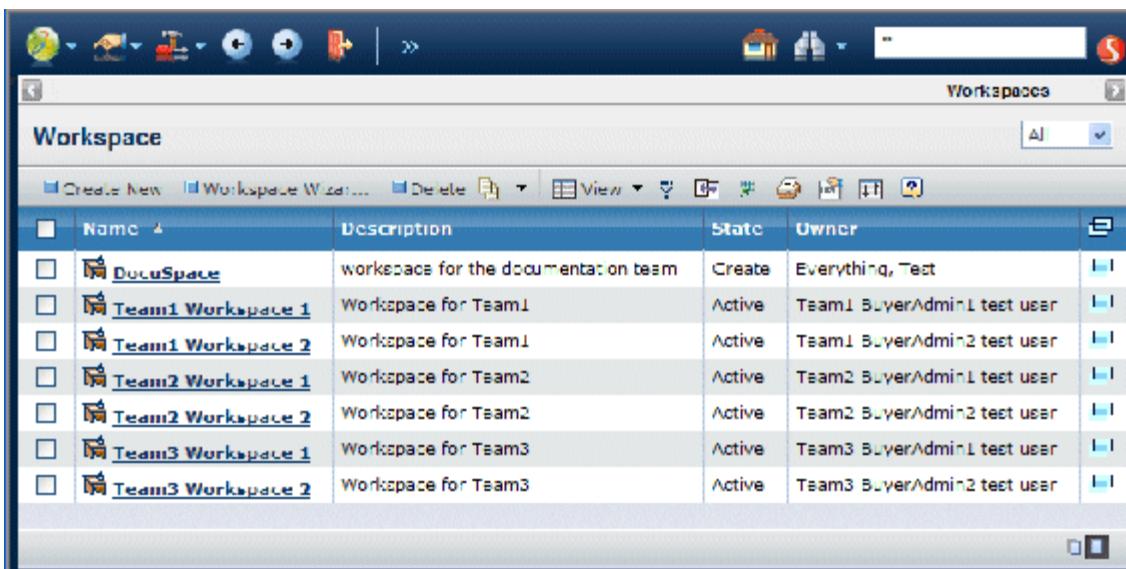
The Navigator page can appear in one of two modes: Menu or Tree. You configure the mode using the mode parameter appended to the URL for the Navigator page. The mode parameter accepts either Menu or Tree as its value. The rest of this section describes these modes and configuration options for them.

The following topics are discussed:

- [Menu Mode](#)
- [Tree Mode](#)

Menu Mode

When the Navigator page is in Menu mode, it contains the global toolbar as it normally does when using the applications. The content frame can contain any application page, such as a table page that lists business objects or a navigation tree and Properties page for one business object. This graphic shows the Navigator page in Menu mode with a table page in the content frame.



Name	Description	State	Owner	Action
DokuSpace	workspace for the documentation team	Create	Everything, Test	[Edit]
Team1 Workspace 1	Workspace for Team1	Active	Team1 BuyerAdmin1 test user	[Edit]
Team1 Workspace 2	Workspace for Team1	Active	Team1 BuyerAdmin2 test user	[Edit]
Team2 Workspace 1	Workspace for Team2	Active	Team2 BuyerAdmin1 test user	[Edit]
Team2 Workspace 2	Workspace for Team2	Active	Team2 BuyerAdmin2 test user	[Edit]
Team3 Workspace 1	Workspace for Team3	Active	Team3 BuyerAdmin1 test user	[Edit]
Team3 Workspace 2	Workspace for Team3	Active	Team3 BuyerAdmin2 test user	[Edit]

The Actions menu items, Create New, Workspace Wizard, and Delete Selected, show as toolbar buttons because of the Toolbar pivot command as defined in "Miscellaneous Properties" in the *Live Collaboration Administrator's Guide*.

To display the Navigator page in Menu mode with any page other than a navigation tree for an object, you specify the page using the ContentPage parameter. For instructions on specifying the content of the Navigator page, see [Configuring the Default Home Page for the Content Frame](#). For example, if the Buyer Desk table page is emxBuyerDeskTable.jsp, the following URL would call the page using the Menu mode.

```
./ematrix/common/emxNavigator.jsp?mode=Menu&ContentPage=.../sourcingcentral/emxBuyerDeskTable.jsp
```

Menu mode is the default mode for the Navigator page so you do not have to specify the parameter if you want Menu mode. To display the Navigator page in Menu mode with the tree and Properties page for a business object in the content frame, include the objectId parameter. For example, to call this page:

The screenshot shows a web browser window with a title bar "Workspaces > DocuSpace : Home". On the left is a sidebar titled "DocuSpace" containing a tree view with three items: "Reports(2)", "Specifications(4)", and "Templates(5)". The main content area is titled "DocuSpace : Folders" and contains a table with the following data:

	Name	Content	Owner	
	Reports	2	Everything, Test	
	Specifications	4	Everything, Test	
	Templates	5	Everything, Test	

Use this URL:

[./ematrix/common/emxNavigator.jsp?objectId=53028.50507.27254.238&mode=Menu](#)



Tree Mode

When the Navigator page is in Tree mode, it does not contain the banner or global toolbar, and the content is *always* the tree for a business object. That is, the default category for that object displays with the Categories menu containing the tree menu for that object type. Use Tree mode when you want to let the user see details for one business object without providing access to other features or applications. Because the Tree mode always displays details for a business object, you must include the objectId parameter whenever you use Tree mode, otherwise the page encounters errors. For example, to show this page:

The screenshot shows a web browser window with a title bar "Workspaces > DocuSpace : Home". On the left is a sidebar titled "DocuSpace" containing a tree view with three items: "Reports(2)", "Specifications(4)", and "Templates(5)". The main content area is titled "DocuSpace : Folders" and contains a table with the following data:

	Name	Content	Owner	
	Reports	2	Everything, Test	
	Specifications	4	Everything, Test	
	Templates	5	Everything, Test	

Use this URL:

[./ematrix/common/emxNavigator.jsp?objectId=53028.50507.27254.238&mode=Tree](#)

Location of the Navigator Page

Like any URL, when the Navigator page executes from a link on a Web page or email message, you use the target href tag to specify where the page should appear: in a new window, in another frame in the current window, or in the current frame.

The following topics are discussed:

- [Portal Parameter](#)
- [Opening the Navigator in Current Window or Frame](#)
- [Examples](#)

Portal Parameter

When opening the Navigator page in a different frame or window, use the portal parameter to make sure the Login page does not replace the frame that contains the link. When the Navigator page executes from within the ENOVIA products and the Login page is required (for example, maybe the system has timed out and a login is required again), the system replaces the current window with the Login page. But if you are calling the Navigator page from an external Web site and want the page to appear in another frame or window, you would typically want the Login page to open in that other frame or window and not within the current frame or window (not within the frame that contains the link).

To ensure the emxNavigator.jsp has no effect on the frame with the link that calls the page, append the portal parameter to emxNavigator.jsp and assign the value of true. If the portal parameter is not included, it is assumed to be false. For example:

```
./ematrix/common/emxNavigator.jsp?portal=true
```



Opening the Navigator in Current Window or Frame

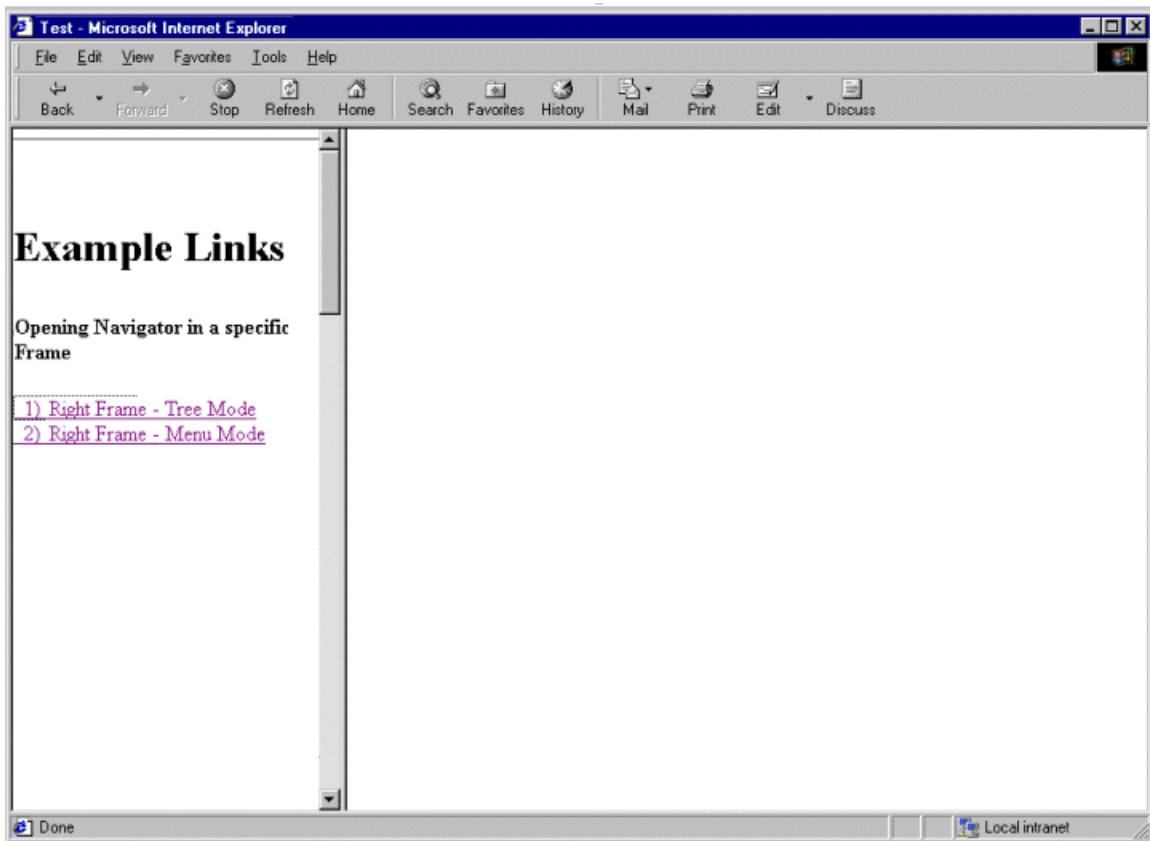
Additionally, to open the Navigator page in a frame within the external Web or ENOVIA portal page that includes the link, include the following bolded code at the top-level page of the implementing external JSP page.

```
*****  
<html>  
<%  
// JSP code here?  
%>  
<head>  
// include START  
  
    <script language="javascript" src="scripts/emxUIConstants.js"></script>  
    <script language="javascript" src="scripts/emxUIObjMgr.js"></script>  
    <script language="javascript" src="scripts/emxUINavbar.js"></script>  
    <script language="javascript" src="scripts/emxUIModal.js"></script>  
    <script language="javascript" src="scripts/emxUITree.js"></script>  
    <script> var tempTree = new jsTree("emxUITree.css"); </script>  
    // include END  
</head>  
<%  
//JSP code here?  
%>  
//*****
```



Examples

For example, suppose you have an external Web page with links to the Navigator page in the left frame, as shown below. Link 1 displays the Navigator page in Tree mode and Link 2 displays it in Menu mode.

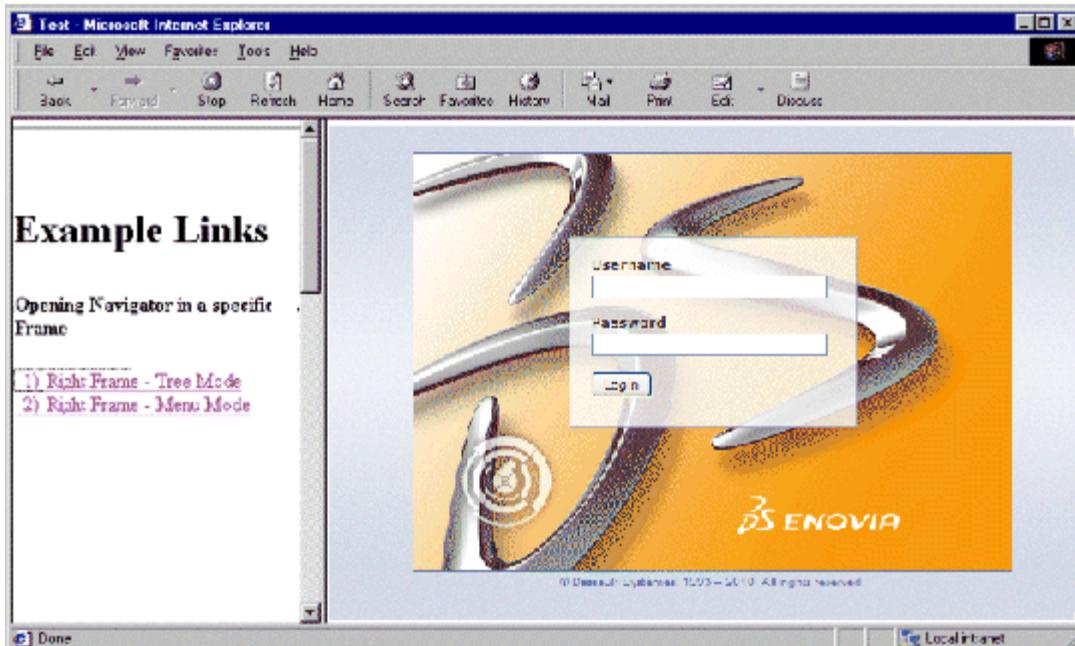


When the user clicks a link, you want the Navigator page to display in the right frame, which is named rightFrame. The href tag for Link 1 would look like this:

```
<a href="emxNavigator.jsp?objectId=53028.50507.27254.238  
&mode=Tree&portal=true" target="rightFrame" >1) Right Frame - Tree Mode </a>
```

Since the Tree mode is specified, the objectId parameter is required so the system knows which business object to show details for. The portal parameter is included because the page should not affect the current frame, the left frame.

If a user clicks Link 1 and is not logged in, the Login page would display in the right frame:



After logging in, the Navigator page opens with the tree and Properties page for the object. Because Tree mode is passed, the page doesn't contain the global toolbar.

Example Links

Opening Navigator in a specific Frame

1) Right Frame - Tree Mode
2) Right Frame - Menu Mode

Part ATPBOM0101 rev 1: Properties

Basic Attributes

Part Mode

Extended Attributes

Name: ATPBOM0101

Part Family:

Design Responsibility:

ECN to release:

State: Approved

Type: Part

Revision: 1

Owner: DesignEngineer, Test

Originated: Jan 31, 2011

Modified: Jan 31, 2011

If Link 2 is configured to display details for the same object but using Menu mode, the href would look like this:

```
<a href="emxNavigator.jsp?objectId=53028.50507.27254.238  
&mode=Menu&portal=true" target="rightFrame" > 2) Right Frame - Menu Mode </a>
```

Assuming the user is logged in, the resulting page would look similar to this:

Example Links

Opening Navigator in a specific Frame

1) Right Frame - Tree Mode
2) Right Frame - Menu Mode

Suppliers

Name	Company ID	Phone Number
Part Store	08 4444 04	617 555 1000
Retro Outfitters	00-4444-02	750-323-1000
Tirestore	05 4444 02	804 898 1000

Forms

You can define form pages using emxForm.jsp and emxFormEdit.jsp.

In this section:

- [!\[\]\(cd7d4b39802967271ef63405c6399431_img.jpg\) About Forms \(emxForm.jsp\)](#)
- [!\[\]\(28d64f1b471c743c9ed28e9c4e991913_img.jpg\) Date/Time Fields in Forms and Tables](#)
- [!\[\]\(58f2d639f215787a80f24c3bb0692ffb_img.jpg\) HTML Components for a Form](#)
- [!\[\]\(25fc352a1b4cdd8677d2e9bb6e7bf6e5_img.jpg\) Building a Form Page](#)
- [!\[\]\(f0e47a412be7c69706ec216832ba1b38_img.jpg\) Form Pre/Post/Cancel Processing](#)
- [!\[\]\(37bbc26e4ad40fbed598951b760b7380_img.jpg\) Additional URL Parameters for Forms](#)
- [!\[\]\(5d59e3600b99faaac53a085ba752a3b1_img.jpg\) Parameters for Web Form Objects](#)
- [!\[\]\(59c090797fe6bb66c9e828f143e80968_img.jpg\) Settings for Fields in Web Form Objects](#)
- [!\[\]\(93f76cc0a8cafbf47b368e728fbcd759_img.jpg\) URL Parameters Accepted by emxForm.jsp](#)
- [!\[\]\(04c5f3f11f36c103752c9d54f7143431_img.jpg\) Actions Menu in a Configurable Form](#)
- [!\[\]\(82ddd0ff77c8cfae47e7676968f5138f_img.jpg\) Filter Toolbar for a Configurable Form](#)
- [!\[\]\(356df661378bc6fcac4607aa472ff8aa_img.jpg\) Design Considerations for Form Fields](#)
- [!\[\]\(845b8d20122208fede456525cbba1076_img.jpg\) Form Fields](#)
- [!\[\]\(fefbf3a36d14032018a30736b1e0b383_img.jpg\) Implementing Range Helpers for Choosers or Custom Pages](#)
- [!\[\]\(ee36bf4cfdbd191cb610e3d9f0ad7d43_img.jpg\) Validating Form Field Data](#)
- [!\[\]\(15bbf94efc32db291e7354587246d6a7_img.jpg\) Committing Changes Made on Edit Form](#)
- [!\[\]\(76616506ddb231e2b36fff4213dc4ab9_img.jpg\) Generic Create Form](#)
- [!\[\]\(1ecdd989b2ff7c241cfe12b2c3ce1517_img.jpg\) JPO Interface for Form Fields](#)

About Forms (emxForm.jsp)

Form pages display the attributes of business objects. The form page lists each attribute in a row, also called a field. Form pages have two modes: View and Edit.

The following topics are discussed:

- [Form Components](#)
- [Links on Form Pages](#)

Form Components

The View form page displays read-only information and can contain an Actions menu that lets users execute actions related to the business object.

The dynamic user interface is designed so the View mode form page for a business object displays when a user views details for the object. The View form displays in the content frame, and the navigation tree for the object displays as the Categories menu.

The screenshot shows a form editor window titled "Part CM-222010-01 rev 1: Properties". The window has a "Page header" at the top. Below it is a "Page toolbar with Actions menu and Categories menu (navigation tree for the object type)". The main area is divided into sections: "Basic Attributes" and "Extended Attributes". The "Basic Attributes" section contains the following fields:

Part Mode	Un configured
Name	CM-222010-01
Part Family	
Design Responsibility	
ECR to release	
Status	Approved
Type	Part
Revision	1
Owner	Design Engineer, Test
Originated	Jan 31, 2011
Modified	Jan 31, 2011
Description	

Annotations on the left side of the screenshot identify parts of the interface:

- "Page header" points to the title bar.
- "Page toolbar with Actions menu and Categories menu (navigation tree for the object type)" points to the toolbar at the top.
- "Form fields" points to the list of attributes in the "Basic Attributes" section.

Forms in edit mode can display in:

- a popup window when the user chooses the Edit toolbar item on the View form's Actions menu
- a slide-in frame
- the content frame itself (the page converts to edit mode instead of opening a new window)

View is the default mode for form pages. To display a form page in Edit mode, add the URL parameter mode=edit to emxForm.jsp. To display specific fields (form rows) as read-only while in Edit mode, use the field setting "Editable".

You can configure the edit link by:

- By passing in a URL parameter called editLink=true. This approach does not support access control.
- By configuring the toolbar command displayed by the form with href assigned to emxFormEdit.jsp. Use this approach when you need access control for the edit command. You must also set the following:
 - Submit=true
 - Popup Modal=true
 - Target Location=popup|slidein|content

Forms in edit mode display in a popup window when the user chooses the Edit toolbar item on the View form's Actions menu.

Links on Form Pages

Forms displayed in View mode can display a toolbar in the header. For instructions on configuring the toolbar menu, see [About Toolbars](#).

The Edit form page always displays Done and Cancel buttons at the bottom.

- The Done button commits the modified attributes to the database. If an exception occurs, the system displays an error message as an alert. The system aborts the transaction if an exception occurs during the update process, and it rolls back all of the updated values.
- The Cancel button aborts any changes and closes the window.

Date/Time Fields in Forms and Tables

Configurable forms and tables support two types of date fields: View mode and Edit mode. To define a date field, use the `format=date` setting. The system always displays dates using the client's locale.

The following topics are discussed:

- [View Mode](#)
- [Edit Mode](#)
- [Date Comparison in Edit Mode](#)

View Mode

In View mode, the date value displays using the `lzDate` taglib. The other settings applicable for date fields in View mode are [Display Format](#) and [Display Time](#). See [Settings for Fields in Web Form Objects](#) and [Settings for Table Column Objects](#).

The Display Date format order of precedence is:

1. The Display Format setting on the form field or table column
2. The property `emxFramework.DateTime.DisplayFormat` defined in `emxSystem.properties`

The Display Time order of precedence for displaying time in date fields is:

1. The Display Time setting on the form field or table column
2. The property `emxFramework.DateTime.DisplayTime` defined in `emxSystem.properties`

If you set the Display Time setting to true, the time display differs based on the Display Format setting and the client's time Zone.

The order of precedence for time zone is:

1. User preference setting for time zone
2. Client's time zone

When the Display Format is either FULL or LONG and Display Time is true, the time shows the client's time zone also. When the user does not set a time zone preference, the display of the time zone includes the Greenwich Mean Time delta (for example, GMT - 05:00) instead of a specific time zone such as EDT. To display a specific time zone such as EDT or PST, users must set a personal time zone preference. If the Display Format is:

Format	Example Time Display
0 - FULL	03:00:00 PM EDT or 03:00:00 PM GMT - 05:00
1 - LONG	03:00:00 PM EDT or 03:00:00 PM GMT - 05:00
2 - MEDIUM	03:00:00 PM
3 - SHORT	03:00 PM



Edit Mode

You can make the date field/column editable by setting `Editable=true`. The Display Format and Display Time settings for date fields in forms or tables are not applicable in Edit mode. The date display uses the Date Format defined by the property `emxFramework.DateTime.DisplayFormat` in `emxSystem.properties`. In Edit mode, Display Time is always False. By default, the text box displays as read-only, but users can change the date by clicking the calendar icon.

When you configure the date field/column in View mode to display the time, and the user changes the date in Edit mode using the calendar icon, switching back to View mode displays the modified date with the time reset to midday (12:00 noon).

Here is the date field of a table in Edit mode with MEDIUM Date Format (specified in the `emxSystem.properties`):

Packed On Date
Sep 1, 2004

Here is the date field of a table in Edit mode with FULL Date Format (specified in the `emxSystem.properties`):

Packed On Date
Monday, January 5, 2004 

Here is the date field of a form in Edit mode with MEDIUM Date Format (specified in the emxSystem.properties):

Modified	Sep 1, 2004 
----------	---

To make a text box (like the one shown above) manually editable, set the form field or table column `Allow Manual Edit = true`.

The order of precedence for making the above text box manually editable is:

- Setting Allow Manual Edit on the form field or table column
- The property `emxFramework.AllowKeyableDates` defined in `emxSystem.properties`.

In a configurable table, the system displays the correct date format under the column header of the field (based on the browser locale). For configurable forms, the system displays it below the text box. This figure shows the date field of a table in Edit mode with MEDIUM Date Format and Allow Manual Edit setting = true in English browser.

Packed On Date (MMM d, yyyy)
Sep 1, 2004 

This figure shows the date field of a table in Edit mode with FULL Date Format and Allow Manual Edit = true in English browser.

Packed On Date (EEE, MMMM d, yyyy)
Sunday, January 11, 2004 

This figure shows the date field of a table in Edit mode with MEDIUM Date Format and Allow Manual Edit = true in French browser.

Date de build planifiée (j nnn aa)
15 sept. 04 

This figure shows the date field of a form in Edit mode with MEDIUM Date Format and Allow Manual Edit = true in English browser.

Modified	Sep 1, 2004 
	(MMM d, yyyy)



Date Comparison in Edit Mode

To support compare date logic for date fields, add a hidden parameter to the web form with the name assigned to the web form field name including the suffix '_msvalue'. If there is a valid display value for the date, the system assigns the hidden parameter with the value that is the equivalent of the displayed date in milliseconds (calculated from midnight, January 1, 1970). The validation methods can then use this hidden parameter value to compare two dates. You can only use this for non-editable date fields.

The hidden parameter gets updated only when you use the out of the box calendar component to change the date. If you change the field type to manual edit, the hidden parameter may not have the updated milliseconds value when the field is manually changed. In this case, the validation method can simply ignore the date compare and must depend on the server side validation. The client side validation can be ignored by checking if the field is read-only.

HTML Components for a Form

The main JSP page in the configurable form component is called emxForm.jsp. This page is the top level frameset and controls whether the displayed page is in View or Edit mode based on the input parameters passed with the URL.

This table lists the frames and forms that make up the View mode of the form page.

Frame Location	Frame Name	Form Name	Includes
header	formViewHeader	--	Page label, Actions menu, page toolbar
body	formViewDisplay	frmFormView	All fields in web form object in read-only display
hidden	formViewHidden	--	--

Building a Form Page

The configurable JSP emxForm.jsp creates form pages. This JSP accepts parameters that define the content and behavior of the page. You pass these parameters to the JSP through the href URL.

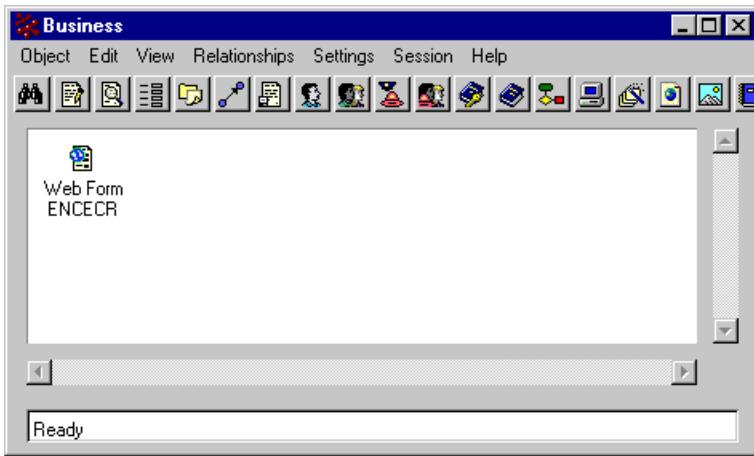
The main components of the form page--the rows of information and the Actions menu (Action menus are for read-only forms)--are defined as administrative objects in the database. You pass the names of these objects to the emxForm.jsp through its parameters.

You must create a web form administrative object for each unique set of fields. For example, suppose the details page for ECRs displays one set of attributes and the details page for parts displays another. You need a web form administrative object for each of these forms, for example CustomECRBasics and CustomPartBasics. Users typically access form pages by selecting an action from an Actions menu, a tree, or tree category. The component that calls the form page must specify the configurable JSP for form pages, emxForm.jsp, in the href parameter. The following procedure lists the main steps for creating a form page and connecting it to an application.

1. Create and configure a web form administrative object and define the form fields to display in the form page. The graphics illustrate the objects needed to create a form page that shows details for a Product Line. The Product Line details page is linked to the tree for Product Lines, menu type_ProductLine.

Compact Copy Machine: Properties	
Categories	
Name	Compact Copy Machine
Primary Image	
Type	Product Line
Description	Includes all Compact Copy Machines
State	Complete
Owner	Design Engineer, Test
Originator	Design Engineer, Test
Program	
Originated	Jan 31, 2011
Modified	Jan 31, 2011
Policy	Product Line
Vault	eService Production
Company	
Marketing Name	
Marketing Text	
Change Board	Sample Change Board 1
Design Responsibility	

When you define form fields, you define the field's data, label, display order, and access. For a description of the available parameters and settings for web forms and fields, see [Parameters for Web Form Objects](#). For naming conventions, see [Naming Conventions for UI Administrative Objects](#).



2. If the form page is in View mode and you want to include an Action menu, configure the action menu and toolbar items using the instructions in [Toolbars](#).

To allow the system to add an Edit link to the top of a View mode form page automatically, include the editLink=true parameter in the href passed to emxForm.jsp. The link calls the same form page but displays it in Edit mode.

3. In the href parameter for the command object that should call the form page (or in the JSP if a JSP is calling the page), enter the URL to display the form page. The URL should include these elements:

- Since emxForm.jsp is in the ematrix/common directory, the URL should begin with the macro for the common directory: \${COMMON_DIR}.
- The URL should include emxForm.jsp.
- The URL should include the parameters needed to display the page, especially the mode parameter that defines whether it should be Edit or View. If the mode parameter is not included, the form defaults to View mode. For a description of the parameters accepted by emxForm.jsp, see [URL Parameters Accepted by emxForm.jsp](#).

For example, the following URL calls the form page in Edit mode:

```
${COMMON_DIR}/emxForm.jsp?form=BuyerDesk&mode=edit
&objectId=3243.32424.232&relId=3432.2342.2344&formHeader
=BuyerDesk&toolbar=SCSBuyerDesktoolbar
```

This URL calls the form in View mode:

```
${COMMON_DIR}/emxForm.jsp?form=BuyerDesk&objectId
=3243.32424.232&relId=3432.2342.2344&formHeader=BuyerDesk
toolbar=SCSBuyerDesktoolbar&editLink=true
```

4. To see your changes while working with the Web-based user interface, select > Utilities > and click the browser Refresh button.

The cache refreshes automatically when the component age expires. The cache refresh setting is in emxSystem.properties. Only the Administration Manager role has access to the Reload Cache tool.

Form Pre/Post/Cancel Processing

When defining a form, you can also define processing to run as the form is being opened (pre), as it is being closed (post), or if the user cancels the operation.

In this section:

- [About Form Processing](#)
- [Pre-Process URL for an Editable Form Page](#)
- [Pre-Process JPO for an Editable Form Page](#)
- [Cancel Process URL for an Editable Form Page](#)
- [Cancel Process JPO for an Editable Form Page](#)
- [Post Process URL for an Editable Form Page](#)
- [Post Process JPO for an Editable Form Page](#)
- [Passing Field Map Information to a Custom JPO](#)
- [Configurable Form View of Multiple Rows and Columns with programHTMLOutput](#)

About Form Processing

You can define values for these URL properties and the associated JPOs or URLs to implement pre-, post-, or cancel processing.

There are 3 types of processing:

- Pre-processing executes before loading the editable form page.
- Post processing executes after completion of the edit process. The edit process, post process JSP, and post process JPO are done in a single transaction. If an exception occurs during the edit or post process, the system rolls back the entire transaction.
- Cancel processing executes whenever a user clicks a Cancel button or closes the editable form component.

To specify a pre/post/cancel processing JSP or JPO, pass the parameters listed in the table below to the appropriate JSP files:

- `emxForm.jsp`
- `emxFormEdit.jsp` (for direct edit modes)

The following parameters are applicable only for edit mode.

URL Parameter	Possible/Default Values	Description
preProcessJPO	<JPOName>:<MethodName>	The JPO to use when a user clicks Edit on a form component.
preProcessURL	<code> \${SUITE_DIR}/emxCustomPreProcess.jsp</code> <i>or</i> <code> ../<ApplicationDirectory>/emxCustomPreProcess.jsp</code> Example: <code>../engineeringcentral/emxCustomPreProcess.jsp</code>	The JSP called before displaying an editable page. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.
postProcessJPO	<JPOName>:<MethodName>	The JPO to use when a user clicks the Done button on a Form.
postProcessURL	<code> \${SUITE_DIR}/emxCustomPostProcess.jsp</code> <i>or</i> <code> ../<ApplicationDirectory>/emxCustomPostProcess.jsp</code> Example: <code>../engineeringcentral/emxCustomPostProcess.jsp</code>	The JSP called when a user clicks the Done button on a Form. The system calls the configured JSP after the edit processing and database update. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.
cancelProcessJPO	<JPOName>:<MethodName>	The JPO to execute when a user clicks the Cancel or close window button on a Form.
cancelProcessURL	<code> \${SUITE_DIR}/emxCustomCancelProcess.jsp</code> <i>or</i> <code> ..<ApplicationDirectory>/emxCustomCancelProcess.jsp</code> Example: <code>../engineeringcentral/emxCustomCancelProcess.jsp</code>	The JSP to call when a user clicks the Cancel button or closes the component window. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.

Pre-Process URL for an Editable Form Page

The preProcessURL parameter specifies the name of the JSP to call during pre-processing of the web form in Edit mode.

Examples to invoke the emxForm.jsp with preProcessURL:

1. Set the href of any command object using macros where appropriate:

```
 ${COMMON_DIR}/emxForm.jsp?mode=Edit&form=<form_name>
 &preProcessURL=${SUITE_DIR}/emxCustonPreProcess.jsp
```

or

2. Invoke emxForm.jsp with custom JSP pages using the relative path (macros are not supported):

```
 ./common/emxForm.jsp?mode=Edit&form=<form_name>
 &preProcessURL=../<Application Directory>/emxCustonPreProcess.jsp
```

If you specify both a preProcessURL and preProcessJPO, the preProcessURL executes first, followed by the preProcessJPO.

The pre-process JSP has these input parameters:

- The request parameters available for the emxForm.jsp, for example, objectId, relId, timeStamp, form, portalMode, suiteKey, etc.
- The request parameter timeStamp gets the form field information stored in the form bean as a map.
- To update or interact with any objects, obtain the context from the request using this code:

```
 context = (matrix.db.Context)request.getAttribute("context");
```

You can read the request parameters in the pre-process JSP page using this code:

```
<%
    // get the timeStamp and objectId from the incoming
    HttpServletRequest
    String timeStamp = (String)
    emxGetParameter(request,"timeStamp");
    String objectId = (String)
    emxGetParameter(request,"objectId");
    String relId = emxGetParameter(request, "relId");
%>
```

To read the form field information in the pre-process JSP page:

1. Define the JSP usebean tag for the formEditBean in the pre-process JSP:

```
<jsp:useBean id="formEditBean"
    class="com.matrixone.apps.framework.ui.UIForm" scope="session"/
    >
```

2. After getting the timeStamp parameter value from the request, get the associated formMap containing the form field details:

```
HashMap formMap = formEditBean.getFormData(timeStamp);
```

3. Retrieve the individual form field information by iterating for each field:

```
MapList formFieldList = (MapList) formMap.get("fields");
for(int i=0; i<formFieldList.length(); i++)
{
    HashMap fieldMap = (HashMap) formFieldList.get(i);
}
```

Pre-Process JPO for an Editable Form Page

You can define a JPO to execute before a form is opened.

The `preProcessJPO` parameter specifies the name of the JPO program and method to execute during pre-processing.

Example to invoke the `emxForm.jsp` with `preProcessJPO`:

```
 ${COMMON_DIR}/emxForm.jsp?mode=Edit&form=<form_name>
 &preProcessJPO=<JPO Name>:<Method Name>
```

If you specify both a `preProcessURL` and `preProcessJPO` are, the `preProcessURL` executes first followed by the `preProcessJPO`.

The pre-process JPO specifies the `programMap` as an argument for the pre-processing. The `programMap` contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains these key parameters and expected values:

Parameter	Description
objectId	Object ID
relId	Relationship ID of the object

- `formMap` contains all the field information

The above maps are packed using the `packArgs` method supported by JPO and passed to the pre-process JPO being invoked. The pre-process JPO can unpack this input parameter and use it for custom coding.

This code samples shows how to get the values for `objectId`, `timeStamp`, `relId`, using a custom processing JPO.

```
// unpack the incoming arguments into a HashMap called 'programMap'
HashMap programMap = (HashMap)JPO.unpackArgs(args);
// get the 'paramMap' HashMap from the programMap
HashMap paramMap = (HashMap) programMap.get("paramMap");
// get the values for objectId, timeStamp, relId, EditAction from the paramMap
String objectId = (String) paramMap.get("objectId");
String timeStamp = (String) paramMap.get("timeStamp");
String relId = (String) paramMap.get("relId");
```

You can retrieve the individual field information by iterating through the `formMap` as shown below.

```
MapList formFieldList = (MapList) formMap.get("fields");
for(int i=0; i<formFieldList.length(); i++)
{
    HashMap fieldMap = (HashMap) formFieldList.get(i);
    String field_expression = (String) fieldMap.get("expression_businessobject");
    String fieldName = (String) fieldMap.get("name");

    //Settings retrieval procedure
    HashMap settingsMap = (HashMap) fieldMap.get("settings");
    String fieldType = (String) settingsMap.get("Field Type");
}
```

The pre-process JPO returns a `HashMap` containing these keys:

Key	Value
Action	Assigned to one of these values: CONTINUE - the process continues STOP - the process stops and the edit dialog closes. Default = CONTINUE.
Message	Either a string resource key or an original text message to display. The string must use proper JavaScript format for escape characters such as single quote, double quote, and new lines.

If the value of the Action key is `CONTINUE`, the system displays the standard edit page. If the value of the Action key is `STOP`, the system does not display the standard edit page.

If the `MESSAGE` key contains a value, the system displays as an alert message on top of the editable page. When the user clicks OK in the message dialog, the editable form page displays (if Action is Continue) or the blank form closes (if Action is Stop).

When the blank window closes (when Action=STOP; MESSAGE contains a value), do not call the onUnload event (cancelProcessJPO or cancelProcessURL).

The following example illustrates a custom pre-processing JPO method that uses most of the pre/post/cancel processing features. See the code comments for details:

```
public HashMap formPreProcObjReserve (Context context, String[] args) throws Exception
{
    // unpack the incoming arguments
    HashMap programMap = (HashMap)JPO.unpackArgs(args);
    // initialize some variables
    String reserveComment = "No Comment Available";
    boolean objIsReserved = true;
    String actionValue = "Stop";

    // get the paramMap, and from it the objectId
    HashMap paramMap = (HashMap) programMap.get("paramMap");
    String objectId = (String) paramMap.get("objectId");
    // get the current user's name
    String currentUser = PersonUtil.getFullName(context);
    String userId = (String) PersonUtil.getPersonProperty(context,"id");
    DomainObject userObject = new DomainObject(userId);
    String userName = userObject.getInfo(context,DomainConstants.SELECT_NAME);
    // build the selects for the current object
    BusinessObjectWithSelect selectList = null;
    StringList objectSelects = new StringList();
    objectSelects.addElement("reserved");
    objectSelects.addElement("reservedby");
    objectSelects.addElement("reservedstart");
    objectSelects.addElement("reservedcomment");
    objectSelects.addElement(DomainConstants.SELECT_TYPE);
    objectSelects.addElement(DomainConstants.SELECT_NAME);
    objectSelects.addElement(DomainConstants.SELECT_REVISION);
    // create a BusinessObject for the current object, open the BusinessObject,
    // get the selects, close the BusinessObject
    BusinessObject busObject = new BusinessObject(objectId);
    busObject.open(context);
    selectList = busObject.select(context,objectSelects);
    busObject.close(context);
    // get the object's reserved status from the select results
    String isReserved = selectList.getSelectData("reserved");
    String reservedBy = selectList.getSelectData("reservedby");
    String reservedStart = selectList.getSelectData("reservedstart");
    String reservedComment = selectList.getSelectData("reservedcomment");
    // get the object's TNR from the select results
    String objType = selectList.getSelectData("type");
    String objName = selectList.getSelectData("name");
    String objRev = selectList.getSelectData("revision");
    // initialize some constants that will be used in the Message string
    String messageString = "Unknown";
    String ALERT_SEPARATOR_STRING = "-----";
    String NEWLINE_CHAR = System.getProperty("line.separator");
    String TAB_CHAR = "\t";
    // set the reserved boolean, depending on whether the object is already reserved
    if(isReserved.equals("TRUE")){
        objIsReserved = true;
    } else {
        objIsReserved = false;
    }
    // if the object is already reserved
    if(objIsReserved){
        // determine if the current user made the reservation. set the Action
        // key accordingly (Continue if this user made the reservation, Stop if not)
        if(reservedBy.equals(userName)){
            actionValue = "Continue";
        } else {
    
```

```

        actionValue = "Stop";
    }
    // build the Message string (all users will get this Message,
    // but the Action key will differ depending on user)
    messageString = "";
    messageString = "----- This object is already Reserved -----";
    messageString += NEWLINE_CHAR;
    messageString += "Name: " + TAB_CHAR + objName;
    messageString += NEWLINE_CHAR;
    messageString += "Type: " + TAB_CHAR + objType;
    messageString += NEWLINE_CHAR;
    messageString += "Rev: " + TAB_CHAR + objRev;
    messageString += NEWLINE_CHAR;
    messageString += ALERT_SEPARATOR_STRING;
    messageString += NEWLINE_CHAR;
    messageString += "Reserved by: " + TAB_CHAR + reservedBy;
    messageString += NEWLINE_CHAR;
    messageString += "When reserved: " + TAB_CHAR + reservedStart;
    messageString += NEWLINE_CHAR;
    messageString += "Comment: " + TAB_CHAR + reservedComment;
    messageString += NEWLINE_CHAR;
    messageString += ALERT_SEPARATOR_STRING;
    messageString += NEWLINE_CHAR;
    messageString += "Current user:" + TAB_CHAR + userName;
    messageString += NEWLINE_CHAR;
    messageString += "\'Action\' key:" + TAB_CHAR + actionValue;
    messageString += NEWLINE_CHAR;
    messageString += ALERT_SEPARATOR_STRING;
    messageString += NEWLINE_CHAR;
}
// if the object is not already reserved
if(!objIsReserved){
    // set the Action key to 'Continue'
    actionValue = "Continue";
    // build a Comment string to include in the reserve operation
    reserveComment = "Reserved by " + userName;
    // open the BusinessObject, reserve the object, close the BusinessObject
    busObject.open(context);
    busObject.reserve(context, reserveComment);
    busObject.close(context);
    // build the Message string
    messageString = "";
    messageString = "----- You have Reserved this object -----";
    messageString += NEWLINE_CHAR;
    messageString += "Name: " + TAB_CHAR + objName;
    messageString += NEWLINE_CHAR;
    messageString += "Type: " + TAB_CHAR + objType;
    messageString += NEWLINE_CHAR;
    messageString += "Rev: " + TAB_CHAR + objRev;
    messageString += NEWLINE_CHAR;
    messageString += ALERT_SEPARATOR_STRING;
    messageString += NEWLINE_CHAR;
    messageString += "\'Action\' key:" + TAB_CHAR + actionValue;
    messageString += NEWLINE_CHAR;
    messageString += ALERT_SEPARATOR_STRING;
    messageString += NEWLINE_CHAR;
}
// build the return HashMap (containing the Action and Message keys)
HashMap returnMap = new HashMap(2);
String actionKey = "Action";
String messageKey = "Message";
returnMap.put(actionKey,actionValue);
returnMap.put(messageKey,messageString);
// send the return HashMap back to the Form component
return returnMap;

```

```
} //end of the formPreProcObjReserve method
```

Cancel Process URL for an Editable Form Page

You can specify a URL to process if the user cancels out of a form.

The `cancelProcessURL` parameter specifies the name of the JSP to call during cancel processing of the web form in edit mode.

Examples to invoke the `emxForm.jsp` with a `cancelProcessURL`:

1. Set the href of any command object using macros where appropriate:

```
 ${COMMON_DIR}/emxForm.jsp?mode>Edit&form=<form_name>
 &cancelProcessURL=${SUITE_DIR}/emxCustonCancelProcess.jsp
```

or

2. Invoke `emxForm.jsp` with custom JSP pages using the relative path (macros are not supported):

```
 ../common/emxForm.jsp?mode>Edit&form=<form_name>
 &cancelProcessURL=..<Application Directory>/emxCustonCancelProcess.jsp
```

If you specify both a `cancelProcessURL` and `cancelProcessJPO`, the `cancelProcessURL` executes first followed by the `cancelProcessJPO`.

The cancel process JSP has these input parameters:

- The request parameters available for the `emxForm.jsp`, for example, `objectId`, `relId`, `timeStamp`, `form`, `portalMode`, `suiteKey`, etc.
- The request parameter `timeStamp` gets the form field information stored in the form bean as a map.
- To update or interact with any objects, obtain the context from the request using this code:

```
context = (matrix.db.Context)request.getAttribute("context");
```

See [Pre-Process URL for an Editable Form Page](#) for codes samples for retrieving request parameters and field information.

Cancel Process JPO for an Editable Form Page

You can define a JPO to execute if the user cancels out of a form.

The `cancelProcessJPO` parameter specifies the name of the JPO program and method to invoke during cancel processing. This example shows how to invoke the `emxForm.jsp` with a `cancelProcessJPO`:

```
 ${COMMON_DIR}/emxForm.jsp?mode>Edit&form=<form_name>
 &cancelProcessJPO=<JPO Name>:<Method Name>
```

If you specify both a `cancelProcessURL` and `cancelProcessJPO`, the `cancelProcessURL` executes first followed by the `cancelProcessJPO`.

The cancel process JPO specifies the `programMap` as an argument for the cancel processing. The `programMap` contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains these key parameters and expected values:

Parameter	Description
objectId	Object ID
rellId	Relationship ID of the object

- `formMap` contains all the field information

The above maps are packed using the `packArgs` method supported by JPO and passed to the cancel process JPO being invoked. The cancel process JPO can unpack this input parameter and use it for custom coding.

You can use the code sample in [Pre-Process JPO for an Editable Form Page](#) to retrieve parameter values and the field information. It includes a code sample that illustrates most of the pre/post/cancel processing features.

The cancel process JPO returns a `HashMap` or nothing. The `HashMap` contains a single key, `Message`, that contains either a string resource key or an original text message to display to the user.

If the cancel Process JPO does not need to communicate to the form, it can return nothing (`void`) or an empty `HashMap`. In this case, the form unloads the page after completing the cancel process.

Post Process URL for an Editable Form Page

You can specify a URL to execute after a form has been closed by the user.

The `postProcessURL` parameter specifies the name of the JSP to call during post processing of the web form in edit mode.

The following examples show how to invoke the `emxForm.jsp` with `postProcessURL`:

- Set the href of any command object using macros where appropriate:

```
 ${COMMON_DIR}/emxForm.jsp?mode>Edit&form=<form_name>
 &postProcessURL=${SUITE_DIR}/emxCustomePostProcess.jsp
```

Or>

- Invoke `emxForm.jsp` with custom JSP pages using the relative path (macros are not supported):

```
 ./common/emxForm.jsp?mode>Edit&form=<form_name>
 &postProcessURL=../<Application Directory>/emxCustomePostProcess.jsp
```

If you specify both a `postProcessURL` and `postProcessJPO`, the `postProcessURL` executes first followed by the `postProcessJPO`.

When the standard edit process completes, the edit display frame `formEditDisplay` is submitted to a hidden target frame with the post process URL. The post process JSP has the following input parameters:

- The request parameters available for the `emxForm.jsp`, for example, `objectId`, `relId`, `timeStamp`, `form`, `portalMode`, `suiteKey`, etc.
- The request parameter `timeStamp` gets the form field information stored in the form bean as a map.
- All the form values displayed in the frame `formEditDisplay` are available to the post process JSP as request parameters.

To update or interact with any objects, obtain the context from the request in the post process JSP using this code:

```
context = (matrix.db.Context)request.getAttribute("context");
```

See [Pre-Process URL for an Editable Form Page](#) for code samples that retrieve request parameters and field information.

Post Process JPO for an Editable Form Page

You can define a JPO to execute after a user closes a form page.

The `postProcessJPO` parameter specifies the name of the JPO program and method to invoke during post processing. This example shows how to invoke the `emxForm.jsp` with `postProcessJPO`:

```
 ${COMMON_DIR}/emxForm.jsp?mode=Edit&form=<form_name>
 &postProcessJPO=<JPO Name>:<Method Name>
```

If you specify both a `postProcessURL` and `postProcessJPO`, the `postProcessURL` executes first followed by the `postProcessJPO`.

The post process JPO specifies the `programMap` as an argument for post processing. The `programMap` contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains key parameters like `objectId`, `relId`

Parameter	Description
<code>objectId</code>	Object ID
<code>relId</code>	Relationship ID of the object
<code>EditAction</code>	Assigned to one of these values: DONE - Default value; edit dialog to be closed APPLY - Edit dialog is still active and the user can continue to edit the same set of objects

- `formMap` contains all the field information

The above maps are packed using the `packArgs` method supported by JPO and passed on to the post process JPO being invoked. The post process JPO can unpack this input parameter and it can be used by the post process for custom coding.

This code sample shows how to get the values for `objectId`, `timeStamp`, `relId`, `EditAction` using a custom processing JPO.

```
// unpack the incoming arguments into a HashMap called 'programMap'
HashMap programMap = (HashMap)JPO.unpackArgs(args);
// get the 'paramMap' HashMap from the programMap
HashMap paramMap = (HashMap) programMap.get("paramMap");
// get the values for objectId, timeStamp, relId, EditAction from the paramMap
String objectId = (String) paramMap.get("objectId");
String timeStamp = (String) paramMap.get("timeStamp");
String relId = (String) paramMap.get("relId");
String EditAction = (String) paramMap.get("EditAction");
```

You can use the code sample in [Pre-Process JPO for an Editable Form Page](#) to retrieve the individual field information. The post process JPO returns a `HashMap` or nothing. The `HashMap` contains the keys defined in this table:

Key	Value
Action	Assigned to one of these values: CONTINUE - commits the transaction STOP - aborts the transaction. Default = CONTINUE.
Message	Either a string resource key or an original text message to display. The string must use proper JavaScript format for escape characters such as single quote, double quote, and new lines.

If the `Message` key contains a value, the system displays it to the user as an alert.

If the value of the `Action` key is `CONTINUE`, the system commits the transaction containing the edit process, post process JSP, and post process JPO.

If the value of the `Action` key is `STOP`, the system aborts the entire transaction, including the edit process.

If the post Process JPO does not need to pass data to the form, it can return nothing (`void`) or an empty `HashMap`. In this case, the process proceeds with the default behavior of Continue and No message.

Passing Field Map Information to a Custom JPO

The form field settings contain information about how the system should display the field, where the field get its data, and how to update it.

The JPOs used with the preProcess JPO, cancelProcess JPO and postProcess JPO parameter get the complete form definition from the key formMap.

fieldMap: fieldMap contains the field settings and the field values based on the type of JPO program executing per the settings defined for the field. The table below describes the list of JPO programs used with the settings, which have fieldMap as an argument.

Field Setting	Description
program	fieldMap is passed for the JPO program. All the keys specified in the fieldMap table below are available to the JPO program.
Range Program	fieldMap is passed to the JPO program. All the keys specified in the fieldMap table below are available to the JPO program.
Update Program	fieldMap is passed to the JPO program. All the keys specified in the fieldMap table below are available to the JPO program.

URL Parameter	Description
postProcessJPO or preProcessJPO or cancelProcessJPO	formMap is passed to the JPO. The formMap contains the form field information (Map) of all the fields in the form. The fieldMap of the individual fields can be retrieved from the formMap. All the keys specified in the fieldMap table below are available to the JPO program.

The fieldMap is a key-value pair and the following table describes its sample structure. Depending on the JPO method getting called, the fieldMap key-value pair available will differ.

Key	Possible or Sample Values	Description
expression_businessobject	relationship[EC Distribution List].to.name	Expression to get the field value from the database
name	ReviewerList	Name of the field
label	emxComponents.Form.Label.ReviewerList	Label given to the field.
Href	URL	Href given to the field.
Settings	Map of all the field settings with the following key-value pairs and the description: Editable - true - Whether the field can be editable Field Type - programHTMLOutput - Determines the field value retrieval procedure. Registered Suite - Components - Application Suite Name	Settings map contains all the information related to the field settings.

This example shows how you can read the field settings of the individual field from the fieldMap.

```
// Getting the fieldMap
HashMap programMap = (HashMap) JPO.unpackArgs(args);
HashMap fieldMap = (HashMap) programMap.get("fieldMap");
// Getting the first-level key-value pair from fieldMap
String field_expression = (String) fieldMap.get("expression_businessobject");

String fieldName = (String) fieldMap.get("name");

String strLabel = (String) fieldMap.get("label");
```

```
//Getting the Settings key-value pair from fieldMap which is inside nested Hashmap  
HashMap settingsMap = (HashMap) fieldMap.get("settings");?  
String suiteKey = (String) settingsMap.get("Registered Suite");?  
String fieldType = (String) settingsMap.get("Field Type");
```

Configurable Form View of Multiple Rows and Columns with programHTMLOutput

If the Field Type is set to programHTMLOutput, you can use the setting Multiple Fields to display multiple rows and columns from a single program html output.

Possible values are `true` or `false`. When `Multiple Fields=false`, output from the program is displayed as a normal programHTMLOutput field.

Setting `Multiple Fields=true` implicitly hides the label and the configurable form does not provide table row and cell html tags while displaying corresponding field values. The programHTMLOutput JPO's must format the field's display and provide the necessary table row and cell html tags.

The `Field Type` must be `programHTMLOutput` and the value form program must be enclosed with `<tr>` tags. This means program html output must provide entire row(s) to display in proper html output.

The system passes an additional value (equates to Total number of table Columns) to the JPO when the `Field Type` is set to `programHTMLOutput`.

If the field count is not known, you can obtain the value using a JPO with the following code:

```
HashMap programMap = (HashMap) JPO.unpackArgs (args);
Integer maxCols = (Integer)programMap("maxCols");
```

Additional URL Parameters for Forms

In addition to directly defining the URL parameters to pass to emxForm.jsp, you can use the appendURL parameter to retrieve additional URL parameters from the properties file for the ENOVIA application.

The `appendURL` parameter takes a value in this format:

`<KEY>|SuiteKey`

Where the `KEY` is a component of a property in the properties file of the specified `SuiteKey`. For example, `appendURL=Effectivity|EngineeringCentral`

The system uses this value, plus the UI component where the `appendURL` is defined (in this case a form), to fetch the list of additional URL parameters to pass to the UI component. The above example corresponds to this property in the `emxEngineeringCentral.properties` file:

```
emxEngineeringCentral.Form.Effectivity =  
appendFields=AdditionalFieldsForm&toolbar={$ORIGINAL},effectivitytoolbar
```

In this example, 2 URL parameters are being returned to the Form page that specified the `appendURL` parameter: `appendFields` and `toolbar`. The `ORIGINAL` macro is used to replace the URL parameter in the original URL string with the one specified in this property. In this example, the toolbar defined in the URL string would be replaced by the one named `effectivitytoolbar`.

As shown by the above example, the `emxSUITENAME.properties` file must contain the required key specified in this format:

`emx<SuiteKey>.<ComponentKey>.<KEY>=VALUE`

where:

- `emx<SuiteKey>` is the application's suite key, such as `emxEngineeringCentral`
- `<ComponentKey>` is the type of UI component requesting the additional URL parameters and is one of these values:
 - Create
 - Form
 - Table
 - StructureBrowser
- `<KEY>` is the specific key being looked up
- `VALUE` is an ampersand-separated list of URL parameters

Parameters for Web Form Objects

This table describes the available parameters for web form objects. The definition of a web form consists primarily of definitions of the fields that make up the form.

All the parameters listed below apply to fields within a form except those that are indicated as applying to the form itself. For specific instructions on how to create web form objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Access (user MQL command)	<p>Use to specify the persons, roles, and groups who can access the field. To make the field available to all users, regardless of role/group assignments, choose All.</p> <p>Note that if no users are assigned access, the system assumes all users have access.</p>	Names of group, role, person administrative objects. or All (default)
Alt	<p>Use to define the ToolTip text to display when users move their mouse pointer over fields configured to display images, or as an alternate to the Label parameter when defining an input control in the page toolbar.</p> <p>Either a string resource ID for the text string or the actual text string. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components.</p> <p>The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text.</p>	emxFramework.Common.ProjectDetails Description
Applies To	<p>Use to specify the item to apply the select expression to: the business object or relationship.</p> <p>Depending on what you specify, the system passes a valid business object ID or relationship ID to emxForm.jsp? For example, emxForm.jsp? objectId=xxx or emxForm.jsp? relId=yyy.</p> <p>Dynamic UI table and tree components pass the objectId and relId automatically to emxForm.jsp, so the href link can be configured without passing IDs.</p>	Business Object Relationship
Description	Use to give a brief description of how the form, field, or input control on a page toolbar should be used. The web form administrative object and each field within it has a description parameter.	Text string
Expression	Use to enter the select expression that gets the field	For business objects: type

	<p data-bbox="297 71 667 333">data. This expression is applied to either the relationship or business object, as specified in the Applies To parameter. If the expression is a business object basic or an attribute that will be editable, make sure you include the Field Type setting with either the basic or attribute value.</p> <p data-bbox="297 344 643 451">The Expression and Applies To parameters are equivalent to the businessobject and relationship MQL commands.</p> <p data-bbox="297 462 643 566">To see an example of a field configured to use an expression, see Field Values as Select Expressions.</p>	<pre data-bbox="715 92 1214 312">name current \$<attribute[attribute_Originator].value> For relationships: \$<attribute[attribute_FindNumber].value> \$<attribute[attribute_Qty].value></pre>
Label	<p data-bbox="297 587 643 798">Use to enter the text that displays as the field's label. Make sure Custom Label is checked so the system gets the label you specify. If Custom Label is unchecked, the system uses the expression as the label.</p> <p data-bbox="297 808 667 988">Either a string resource ID for the text string or the actual text string. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components.</p> <p data-bbox="297 998 643 1157">The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it does not find one, it displays the entered text.</p>	<pre data-bbox="715 587 1183 692">emxEngineeringCentral.common.Name emxTeam.Common.ProjectName Description</pre>
href	<p data-bbox="297 1189 643 1358">Use to specify the URL that gets executed when the field data is clicked. This parameter is used for View mode. In Edit mode, it is ignored because fields in Edit mode are not shown with a hyperlink.</p> <p data-bbox="297 1368 643 1685">When a user clicks the hyperlinked field data, the system passes the objectId parameter as part of the URL. By default, the value for the objectId parameter is the ID of the business object the form page applies to. Using the Alternate OID expression setting, you can configure the field to pass the ID returned from a different business object.</p> <p data-bbox="297 1695 643 2008">Also used for an input control in the page toolbar to specify the URL to execute when a user clicks the submit button. The <code>Input Type = combobox</code> also executes the href on the <code>onChange</code> event and the <code>Input Type = checkbox</code> executes the href on the <code>onClick</code> event. You also need to provide an Action Label setting on the input control to actually draw the</p>	<pre data-bbox="715 1189 1190 1252"> \${COMMON_DIR}/emxTree.jsp \${SUITE_DIR}/emxpathEditPartDialog.jsp</pre> <p data-bbox="715 1262 2259 1315">To see an example of a field that uses an href, see Field Values as a Hyperlink and Type Icon, Field Values with Hyperlinked Data Using an Alternate OID and Alternate Type Icon, and Field Values as Hyperlinked Image.</p>

	<p>submit button (Action Label defines the name that shows on the submit button).</p> <p>The value for the href parameter should be a JSP and any associated parameters. You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components.</p>	
Name	<p>Use to enter the name of the field used as identifier for the field within the web form object.</p> <p>This parameter is also used to define the name of an Input Control for the toolbar.</p>	Name Revision Material Category
Name (of web form)	<p>Use to enter the name of the web form administrative object. You must enter the actual name of the web form; symbolic names are not supported. For naming conventions, see Naming Conventions for UI Administrative Objects.</p>	ENCPart
RangeHref (range MQL command)	<p>When configuring a form field in Business Modeler, the RangeHREF parameter is set on the Link tab.</p> <p>Use to configure a textbox that has a Browse (...) button that calls a chooser or custom window from which users can select a value to populate the textbox. This Owner textbox is an example with a range helper.</p>  <p>In the RangeHref parameter, specify the href URL to display the window. For example, the href might call a custom selection page.</p> <p>Range helpers are available only in Edit mode. Only textbox controls can be configured with a range helper. To specify the control type, set Input Type=textbox.</p> <p>To see an example of a field configured with a RangeHref, see Field with Popup Range Helper. Also see Implementing Range Helpers for Choosers or Custom Pages.</p>	<p>You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components. For example:</p> <pre> \${COMMON_DIR}/emxSelectVault.jsp \${SUITE_DIR}/emxSelectUser.jsp emxTypeChooser.jsp?typeList=type_Part,type_Document ./common/emxTypeChooser.jsp? &SelectType=multiselect&SelectAbstractTypes=true&InclusionList=eServiceEngineeringCentral.Types&observeHidden=true&>ShowIcons=true </pre>
renderPDF	<p>Parameter to control whether or not to show the Render PDF icon in the form toolbar when displaying the form in view mode.</p> <p>If the parameter is not passed in, false is assumed and the icon is not displayed.</p>	false (default) true

Settings	Additional settings that define the behavior and appearance of the column.	Name/value pairs, as defined in Settings for Fields in Web Form Objects .
Type (for web form)	Dynamic user interface components do not use the type parameter.	--
Update URL	Dynamic user interface components do not use this parameter. They use the Update Program and Update Function settings instead.	--

Settings for Fields in Web Form Objects

This table lists and describes the settings for web form objects. The name and value for each setting are case sensitive.

Setting	Description	Accepted Values/Examples
*Registered Suite	The application the field belongs to. The system looks for files related to the field in the registered directory for that application, which is specified in emxSystem.properties. Based on the application name, the system passes the following parameters in the href URL: suiteKey emxSuiteDirectory StringResourceId	Set the value without any spaces, for example, EngineeringCentral or Framework. Set the value to the suite name as defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. If the suite name starts with eServiceSuite then you can skip this prefix and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is eServiceSuiteEngineeringCentral, then the word EngineeringCentral, can be assigned as "Registered Suite".
Access Expression	Used to control access to form fields For details, see User Access to UI Components .	--
Access Function		
Access Mask		
Access Program		
Additional Query	When Type Ahead is configured on the field (by setting the RangeHref to emxFullSearch.jsp or using a predefined Type Ahead Chooser), this setting defines additional field/selection critiera used to restrict the selection list. See About Automatic Type Ahead .	<FieldName1>=<select expression1>: <FieldName2>=<select expression2>:
Admin Type	Use to translate fields whose values are names of administrative objects or ranges of attributes. For example, suppose you are configuring a field that shows an object's current state and you want the state name to be translated. You would add this setting and set the value to State. The translations for administrative object names are stored in the emxFrameworkStringResource.properties files, as described in Internationalizing Dynamic UI Components .	These keywords get the field values translated for the appropriate type name: <ul style="list-style-type: none">• Type• State• Role• Relationship• Policy• Group• Vault• Attribute (for translating the attribute name, not the range values) To translate attribute and range values, specify the symbolic name of the attribute, which starts with "attribute_" and is followed by the attribute name with no spaces. For example: <ul style="list-style-type: none">• attribute_UnitOfMeasure• attribute_PartClassification
Allow Manual Edit	When true, users can manually edit the form row for this field. Applicable only when the range parameter is set to a URL or when the setting format is assigned to date or for fields of type combobox. It is ignored in all other cases. When this setting is true, the Admin Type setting is ignored.	false (default) --Manual entry is not allowed. true--Manual entry is allowed.
Alternate OID expression	By default, when a field's data is configured to show as a hyperlink using the href parameter, the system passes the ID for the business object the form page applies to. Using this setting, you can have the system pass the ID(s) for a different object, namely, the ID(s) for the object(s) returned from the expression specified in this setting.	\$<to[relationship_NewPartPartRevision].from.id> \$<to[relationship_EBOM].from.id> To see an example of a field that uses an Alternate OID expression and an Alternate Type expression, see Field Values with Hyperlinked Data Using an Alternate OID and Alternate Type Icon .
Alternate Type expression	When the Show Alternate Icon setting is true, this expression is used to obtain the object type. Based on	\$<to[relationship_NewPartPartRevision].from.type> \$<to[relationship_EBOM].from.type>

	the obtained type, the corresponding icon is displayed.	
Calendar Function	Use to specify the name of the method in the JPO specified in the Calendar Program setting that retrieves the non-working days based on the calendar defined for the location.	The name of a function in the Calendar Program JPO, such as: <code>getNonWorkingDays</code>
Calendar Program	Use to specify the name of a JPO that contains a method to get the non-working days for a calendar.	The name of a JPO, such as: <code>emxWorkCalendar</code>
Category	Used to select attributes with the same <code>uiform_Category</code> property setting.	For example: D2MBusiness EC Technical
Cols	Used when the input type is set to <code>textarea</code> . This setting limits the length of the <code>textarea</code> on the form and specifies the visible width in average character widths. If not specified, it uses the HTML default, which is 25.	25 40 50
Column Count	The number of name/value columns to draw horizontally.	1 (default) 2 3 <i>n</i>
Create Exclude	Comma-separated list of attributes that will not be included in the webform when opened in Create mode when a field on the webform has been defined as Field Type = Dynamic Attributes.	For example: <code>attribute_Cost</code>
Decimal Precision	Defines the decimal precision for all calculations for this form. The system properties setting <code>emxFramework.FormCalculations.DecimalPrecision</code> defines the system-wide value; this setting overrides that value. You only need to use this setting if you want to use a value other than the system-wide value (defined in <code>emxSystem.properties</code>).	Any positive integer.
Default	If a field's value is empty or null and this setting is defined, the default value is displayed for the field.	The default value you want to display. This can be a string resource key or the actual characters you want to fill in as the default. The wildcard (*) can be used for search criteria fields. <code>emxFramework.Common.default</code> All *
Delimiter	When using <code>Input Type = dynamictextarea</code> setting, this setting defines the character that separates values when the field is in View mode. In Edit mode, each value shows on a separate line. A comma is the default delimiter. If you specify any of these characters as the delimiter, a comma is used instead: <code>\$ \ ' " * ? () ></code>	<any string value> , (default)
Display Format	Specifies the number of the date format for any field where the value of the format setting is "date." See Date/Time Fields in Forms and Tables .	These are Java standard values of Date Format to display a date in a specific format. 3 - SHORT (12/12/52) 2 - MEDIUM (Dec 12, 1952) 1 - LONG (December 12, 1952) 0 - FULL (Tuesday, December 12 1952 AD) Default is set in <code>emxSystem.properties</code> : <code>emxFramework.DateTime.DisplayFormat=MEDIUM</code> . <code>emxSystem.properties</code> uses words, but the Display Format setting uses numbers.
Display Time	Controls whether the time displays with the date for fields whose format is set to date.	true

	If no time zone preference is set, then the DateTime is shown in the browser's time zone. The time is shown in terms of GMT+/- hh:mm, (e.g., Saturday, August 21, 2004 12:45:00 PM GMT-04:00). To get the time in a format like EST or PDT, set the time zone preference to a specific zone. See Date/Time Fields in Forms and Tables .	false Default is set in emxSystem.properties for the property emxFramework.DateTime.DisplayTime = false
findMxLink	When true, show the mxLink icon/command button on the toolbar, which opens a search dialog box.	true (default) false
Editable	Use to indicate whether the field is displayed as editable or read only. Only applies for Edit mode. View mode ignores the setting.	true (default)--Users can edit the field when shown on the Edit mode form. false--Users cannot edit the field when shown on the Edit mode form. The field looks just like it does in View mode except it is never hyperlinked.
Edit Exclude	Comma-separated list of attributes that will not be included in the webform when opened in Edit mode when a field on the webform has been defined as Field Type = Dynamic Attributes.	For example: attribute_Cost
Export	Specifies whether field data is exported to csv or not. Use this to change the export value on a field-by-field basis.	true false
Field Column Headers	Used in conjunction with the Field Type=Table Holder setting. Specifies the labels for the column headings. The number of labels should be the same as the value for the Field Table Columns setting and should be separated by a comma. You can specify either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. See Vertical Grouping .	Comma-separated list of column heading labels: Min,Max,Avg
Field Row Headers	Used in conjunction with the Field Type=Table Holder setting. Specifies the labels for the row headings. The number of labels should be the same as the value for the Field Table Rows setting and should be separated by a comma. You can specify either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. See Vertical Grouping .	Comma-separated list of row heading labels: Weight,Volume
Field Size	Determines the width of a textbox input type field. The width is given in pixels except when Input Type is textbox or not set. In that case, its value refers to the (integer) number of characters.	Number of pixels, for example: 30 20 is the default.
Field Table Columns	Used in conjunction with the Field Type=Table Holder setting. Defines the number of columns for the table. See Vertical Grouping .	2, 3, ...
Field Table Rows	Used in conjunction with the Field Type=Table Holder setting. Defines the number of rows in the table. See Vertical Grouping .	1, 2, ...
Field Type	This setting is used: <ul style="list-style-type: none"> To indicate that the field's data should be obtained from a program or image instead of an expression. When the field data is obtained from an expression, if the data is basic information or an attribute. The system needs to know whether a field's data is basic information or an attribute in order to update the information correctly. Specifying whether the field is basic or an attribute is only required for 	program--The values for this field are obtained from a program (JPO). The program and function name are required as settings. programHTMLOutput--Same as the program setting, except the field value output is in HTML format. Field values are placed in the table cell between <td> and </td> tags. This setting ignores other field settings such as Show Type Icon, href, format, and Alternate OID expression.

	<p>fields that will be editable.</p> <ul style="list-style-type: none"> To indicate the field is a dummy field that defines fields to display in a table or group. <p>For more information on specifying field data, see Form Fields.</p> <p>The Group Holder setting has been deprecated.</p>	<p>Dynamic Attributes--Displays all attribute/value pairs associated with the context object in the properties page.</p> <p>ClassificationPaths--Used only with Library Central. Displays the paths where the object is classified, with a separator between hierarchies. This separator is configurable with the Library Central property string emxLibraryCentral.ClassPathSeparatorString. The default separator is '----'.</p> <p>ClassificationAttributes--Used only with Library Central and the Multiple Classification Module. Displays the attributes acquired via classification. If the object is classified, it displays the classification name as the heading, and then displays a subheading with the name of the attribute group, and then attributes and their values acquired from the attribute group.</p> <p>If the same attribute is repeated in another attribute group or in another classification, this field displays a message underneath the field: "This value also appears in another Attribute Group." If a user modifies one attribute and has other occurrences in different attribute groups, this automatically updates all other occurrences of the attribute. If the object is classified but no attributes are acquired via classification, this field does not display anything.</p> <p>image--The field's value is the primary image associated with the business object.</p> <p>basic--The field displays basic information for the business object. Basic information includes name, type, originated, policy, etc. Specifying basic as the field type is only needed when the field is editable. The only editable basic information is: type, name, revision, current, policy, description, owner, vault.</p> <p>attribute--The field displays values for an attribute on the business object, such as Originator or Weight.</p> <p>Section Header--Adds a new section heading between the form fields. The setting Section Level determines the heading level. See Field as Section Header and Separator.</p> <p>Section Separator--Adds white space to separate fields and sections.</p> <p>Table Holder--Arranges the fields under the field in columns and rows. Table Holder fields serve as dummy fields to define the fields to display in a table. Also see these settings: Field Column Headers, Field Row Headers, Field Table Columns, Field Table Rows.</p> <p>emxTable--Embeds a configurable table in the form. Used in conjunction with the table setting and either the inquiry or program setting. See Field that Embeds a Configurable Table.</p>
format	<p>Use to specify the type of data in the field. If the Editable setting is true and the field is on an Edit mode form, the system uses these format values to validate the field value. Validation takes place on the client side, before updating the object displayed in the form. For more information on validation, see Validating Form Field Data.</p> <p>To support the date compare logic, whenever the field format is "date", an additional hidden parameter is added to the form with the name assigned to the web form field name suffixed by "_msvalue". If there is a valid display value for the date, the hidden parameter is assigned with the value that is the equivalent of displayed date in milliseconds (calculated from midnight, January 1, 1970).</p>	<p>date--Uses the tag lib to format the displayed field value based on the browser locale setting. To see an example of a field configured as a date field and that has date validation, see Field Value with Dates.</p> <p>currency</p> <p>numeric--Use if the value must be validated as a number before updating the values. Applicable only in Edit mode.</p> <p>email--Displays the column values as an email address. When a user clicks the email address, the email editor configured in the client is presented.</p>

	<p>This hidden parameter value can be used by the validation methods to compare two dates.</p> <p>Note: The hidden parameter gets updated only when the out-of-the-box calendar component is used to change the date. So if the field type is changed to manual edit, the hidden parameter may not have the updated milliseconds value when the field is manually changed. In this case the validation method can simply ignore the date compare and must depend on the server side validation. The client side validation can be ignored by checking if the field is readonly.</p>	
function	<p>The name of the method to call within the JPO program specified in the program setting. This method within the JPO is used to get the field values if the setting "Field Type" is set to "program" or "programHTMLOutput".</p> <p>For more information, see Field Values Obtained from a Program.</p>	<p>The name of a function in the program JPO, such as:</p> <p>getAssignedBuyerDesk getPackageAccess getParentPart</p>
Group Name	<p>Used for grouping fields in web forms. The consecutive fields with same group name are considered a group. See Fields that are Grouped.</p>	<p>The name of the group.</p>
Help Marker	<p>Specifies the name of the help marker to call for context-sensitive help.</p> <p>In the href URL called when the field data is clicked, the system passes a parameter called HelpMarker and includes the marker text specified for this setting.</p>	<p>The naming convention for help markers is the page title, as displayed at the top of the visual page, prefixed with "emxhelp". The marker is all lowercase with no spaces.</p>
Hide Label	<p>Displays or hides the label for a particular row on a form.</p>	<p>True False</p>
Image	<p>Use to specify an image file when the field value should be an image only. This setting is required when the Field Type setting is set to image. This file must exist in the application server (not in the database). You can make the image a hyperlink by including a URL in the href parameter.</p> <p>To see an example of a field with an image, see Field Values as Hyperlinked Image.</p>	<p>images/newPart.gif images/EditItem.gif</p>
Image Size	<p>When the web form includes a Field Type of image, this setting defines which size of the primary image associated with the business object should display in the column. The pixel dimensions for these sizes are defined using the emxComponents.Image.SizeType property (see the <i>Live Collaboration Administrator's Guide</i>).</p>	<p>format_mxSmallImage (default) format_mxLargeImage format_mxMediumImage format_mxThumbnailImage</p>
Input Type	<p>Only used for forms and table columns in edit mode. Specifies the type of HTML control to display for user input.</p> <p>Although multiple choices can be displayed in a webform, only one selection can be saved during edit. If you want to disable the ability to make multiple selections during view of webform, you need to use programHTMLOutput and the html tag that does that.</p> <p>Use the radiobutton or combobox input types for fields that require a single selection.</p> <p>If you want to save multiple selections, a custom JPO needs to be written using the checkbox or listbox input types which might delimit the choices in the attribute using the Studio Customization Toolkit.</p> <p>If you specify checkbox or radiobutton for an attribute with the String datatype, the attribute must be defined with a range or an error occurs when the form is in Edit mode.</p> <p>If using dynamictextarea, the Delimiter setting also needs to be defined if you want to use a delimiter</p>	<p>textbox-Default. Provides a single-line box for typing text. This graphic shows a field configured as a Text Box with the Required setting equal to true. If the attribute is configured with a dimension, a drop-down list of units displays after the test box</p>  <p>textarea-Provides a multi-line box for typing text.</p>  <p>checkbox-Provides a check box next to each range value.</p>  <p>See Sample JPO for Web Form with Custom Combobox.</p> <p>listbox-Provides a list of range values. Although users</p>

	<p>other than the default (comma).</p>	<p>can select more than one item in the list by holding the Ctrl key, only one value will be saved unless a custom JPO is implemented.</p>
	<p>Unit of Measure</p>  <ul style="list-style-type: none"> <input type="radio"/> LB (pound) <input type="radio"/> IN (inch) <input type="radio"/> GA (gallon) <input type="radio"/> FT (feet) <input checked="" type="radio"/> EA (each) 	<p>radiobutton-Shows a radio button next to each range value. To see an example of a field configured with radio buttons, see Field for Attribute with Choices in Radio Buttons.</p>
		<p>combobox-Provides a drop-down list of options and users can only select one value. Use combo boxes for attributes that have defined ranges and for attributes whose ranges are determined with a Range Helper URL. To see an example of a field with a combo box, see Field for Attribute with Choices in Combo Box.</p> <p>ECR Evaluator <input type="text" value="Test ECREvaluator"/> </p>
inquiry	<p>Used only for fields defined with Field Type=emxTable. Either this setting or the program setting must be used to define the objects to retrieve. Specifies the inquiry administrative object that should be used to retrieve the business objects to be included in the table.</p>	<p>Name of inquiry administrative object. inquiry=SCSBuyerDesk inquiry=ENCAIIIParts</p>
Label	<p>Use to enter the text that should appear as the label for the field. Make sure Custom Label is checked so the system gets the label you specify. If Custom Label is unchecked, the system uses the expression as the label.</p> <p>Either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components.</p> <p>The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it does not find one, it displays the entered text.</p> <p>When used with a field type of Dynamic Attributes, the Label becomes the section heading name, or if not provided or set to blank, indicates that no section heading will be displayed.</p>	<p>emxEngineeringCentral.common.Name emxTeam.Common.ProjectName Description</p>
Maximum Length	Limits the number of characters that can be entered in a field with Input Type of textbox. If not specified, it uses the HTML default (unlimited).	Number of characters, for example: 30
OnChange Handler	The name of the JavaScript function called when the value in the field is changed. You can provide a semicolon separated list of JavaScript functions.	<JavaScript function name>
OnFocus Handler	The name of the JavaScript function called when a field is selected for edit. You can provide a semicolon separated list of JavaScript functions.	<JavaScript function name>
Popup Modal	<p>Specifies whether windows opened from a link are modal or non-modal.</p> <p>Used when the setting Target Location is set to popup.</p> <p>If a form Component is opened in a dialog in edit mode then the dialog must open as a modal dialog. For example, if a table column is designed to pop up the Form Component in edit mode, then that column must have a setting Popup Modal=true to open the dialog as</p>	<p>true--the popup window is modal false--the popup window is non-modal If the setting is not specified, the window is modal.</p>

	modal dialog.	
Printer Friendly	Specifies whether the target page should include the Printer Friendly tool. In the href URL called when the field data is clicked, the system passes a parameter called PrinterFriendly and includes the value specified for this setting. If this setting is not included, the value is assumed to be false.	True False (default)
program	<p>Use to specify the name of the JPO program to get the field's data. This program gets the field values when the Field Type setting is <code>program</code> or <code>programHTMLOutput</code>.</p> <p>Can be used for fields defined with Field Type=emxTable to define the objects in the table. Alternatively, inquiry can be used.</p> <p>Using a JPO to populate field data is recommended only when a select expression cannot be used to obtain the field value.</p> <p>To see an example of a field that uses a program, see Field Values Obtained from a Program.</p>	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
Range Function	Use to specify the name of the method in the JPO specified in the Range Program setting. See the Range Program setting below for more information.	The name of a function in the Range Program JPO, such as: getAssignedRange getClassificationRange getPartUOM
Range Program	<p>Use to specify the name of a JPO that contains a method to get the field value ranges (choices). A range program is used only when the Input Type setting is combobox, radiobutton, or checkbox.</p> <p>If the choices need to be presented on a page in a popup window, for example in a chooser, use the RangeHref parameter instead.</p> <p>To see an example of a field configured with a range program, see Field for Attribute with Choices in Combo Box. Also see JPO for Getting Field Range Values (Choices).</p>	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
Read Only Checkbox	<p>For a field defined with the Boolean data type, or as a string with Boolean range values, this setting shows a checkbox instead of the TRUE or FALSE text.</p> <p>In edit mode, the user can check/clear the checkbox unless the Editable=false setting is specified for the field.</p>	true false (default)
Reload Function	<p>The name of the method in the JPO specified by the Reload Program setting that executes a field reload.</p> <p>You also need to specify a value for the Reload Program setting.</p>	<JPO method name>
Reload Program	<p>The name of a JPO to invoke when this code in the form is called:</p> <pre>emxFrmReloadField("FieldName")</pre> <p>You also need to specify a value for the Reload Function setting.</p>	<JPO Name>
Remove Range Blank	Used when the <code>input type</code> is set to <code>checkbox</code> to ensure that the field contains a value and is not left blank. If set to True, this setting removes the blank from the combo box list. If a default is not specified, the first value in the list is shown by default.	true false (default)
Required	Use to indicate the value for this field is required. This setting is applicable for editable fields displayed on the Editable mode form.	true--When completing the form, the user must enter a value for the field. The field label appears in red italic text. If the user does not enter a value and clicks Done, a JavaScript message appears that prompts the user to enter a value.

		false (default)--When completing the Edit form, the user can leave this field blank.
Rows	Used when the <code>input type</code> is set to <code>textarea</code> . This setting limits the height of the textarea on the form and specifies the number of visible text lines. If not specified, it uses the HTML default, which is 5.	5 10 20
Section Level	Used in conjunction with the Field Type: Section Header setting to define the level of heading. To see an example, see Field as Section Header and Separator .	Two heading levels are available: 1 (default)--Font for heading label is large and a horizontal line is included above the label. 2--Font for heading label is smaller and the heading is in the same gray rectangle as standard fields.
Show Alternate Icon	Set to true if the field value must display with an icon other than the current object type icon. To get the right alternate icon, you must define the Alternate Type expression with the expression to obtain the object type.	true false (default)
Show Clear Button	Adds a Clear hyperlink next to the field. This setting only applies to textarea/textbox input type fields. When the user clicks the Clear link, it clears the content of the textbox/textarea input type field.	true false
Show Type Icon	If set to true, the field value displays along with the type icon for the business object being displayed by the current form page. The icon is defined in the <code>emxFramework.smallIcon</code> property in <code>emxSystem.properties</code> . The icon displays to the left of the field data. If no icon is defined for this type, the system looks for a property defined for the parent type and so on up the hierarchy. If no property is defined for any type in the hierarchy, the system uses the default icon specified in the <code>emxFramework.smallIcon.defaultType</code> property.	true false (default) To see an example of a field with a type icon, see Field Values as a Hyperlink and Type Icon .
sortColumnName	Used only for fields defined with Field Type= <code>emxTable</code> . Specifies the column the table should sort on when the page is first loaded. If no column is specified, the rows are listed in the order they are retrieved from the database.	Name of column in the table specified in the <code>table=TABLE_NAME</code> setting.
Sort Direction	Used only for fields of type combobox and listbox (attributes with defined ranges). The sort order for the option list.	ascending (default)--Sort a to z or 0 to n. descending--Sort z to a or n to 0. none--The option list is not sorted.
Sort Range Values	Enables or disables the sorting of range values in combobox or listbox controls. When enabled, the list is sorted based on the datatype and using the direction defined by the Sort Direction setting. When disabled, no sorting is done on the list. In a drop-down, range values populated based on an attribute expression will be sorted based on the attribute's datatype. Range values populated using the Range Program and Range Function settings will be sorted alphanumerically. If the Range Program or Range Function returns numeric or date values, the alphanumeric sort will not be appropriate. The program or function should sort the values in the required order, and this setting should be disabled. This setting does not apply to fields or columns configured for attributes associated with a dimension. Dimension ranges are always sorted as alphanumeric in ascending order. This setting cannot be used in custom JSPs that use the <code>editOptionList</code> taglib. For this specific situation, you can use the <code>sortType</code> attribute for that tag.	enable (default) disable

table	<p>Specifies the name of a table administrative object to embed in the form. Used only for fields defined with Field Type=emxTable. You must use the actual table name; symbolic names are not supported. See Field that Embeds a Configurable Table.</p>	<p>Name of table administrative object. For example: table=SCSBuyerDesk table=ENCParts</p>
Target Location	<p>Controls where the page specified in the href parameter appears when a user clicks the hyperlinked data.</p> <p>To specify a tab in a PowerView window, use the administrative command name that configures the tab in the PowerView page. If the specified command name is not defined within the current PowerView, the popup value is used.</p> <p>When using slidein as the Target Location and the Popup Modal setting is true, then the content window and the global toolbar are grayed out and disabled until the slidein window is closed.</p>	<p>popup--Page appears in a new window. The window modality can be set with the Popup Modal setting. Popup is the default value and is used if the setting is not included or if the named frame cannot be found.</p> <p>content--The page replaces the content frame.</p> <p>mainFrame--Page appears in the frame that includes the content and menu frames.</p> <p>_top--Page replaces the entire body of the browser window.</p> <p><command name>--The name of a command configured as the target tab in a PowerView page.</p> <p>slidein--Page appears in a slidein frame (slides in along the right side of the window).</p> <p>This value can be set to any valid frame name that is available to the form body frame.</p>
Tip Page	<p>Specifies whether the target page should include the Tip Page tool (light bulb icon). In the href URL called when the field data is clicked, the system passes a parameter called TipPage and includes the URL specified for this setting. If this setting is not included, the value is assumed to be false and the target page does not display the Tip Page tool.</p>	<p>Name of a custom URL page.</p>
Type Ahead Mapping	<p>When Type Ahead is configured on the field (by setting the RangeHref to emxFullSearch.jsp or using a predefined Type Ahead Chooser), this setting maps that field to the corresponding indexed field in config.xml. See About Automatic Type Ahead.</p>	<p>NAME LASTNAME,FIRSTNAME,USERNAME</p>
Type Ahead Validate	<p>When Type Ahead is configured on the field, this setting determines if the application should restrict the user to selecting one of the suggested values (if true), or if the user can enter any data in the field without BPS validating that data (if false).</p>	<p>true false (default)</p>
Type Icon Function	<p>Specifies the method in the JPO specified in the Type Icon Program setting that retrieves an icon to show in addition to the Alternate Icon or Type Icon (enabled by Show Alternate Icon or Show Type Icon settings).</p> <p>If both the Show Alternate Icon and the Show Type Icon settings are set to false, any value for this setting is ignored.</p>	<p>Method Name</p>
Type Icon Program	<p>Defines the program that contains the function specified using the Type Icon Function setting.</p>	<p>JPO Name</p>
TypeAhead	<p>Enables type ahead. If not provided, automatically set to true. Typically used to disable type ahead for a field.</p>	<p>true false</p>
TypeAhead Program	<p>Name of the JPO called to retrieve a list of possible values. This JPO is called after the user types the specified number of characters. A JPO is typically used by fields that are normally populated using choosers.</p>	
TypeAhead Function	<p>The name of the JPO method used in conjunction with the TypeAhead.Program setting.</p>	
TypeAhead Character Count	<p>Overrides the emxFramework.TypeAhead.RunProgram.CharacterCount system property. This setting may be useful for fields whose values may all contain the same prefix.</p>	<p>2 (default) 5</p>
TypeAhead Saved Values Limit	<p>Overrides the emxFramework.TypeAhead.SavedValuesLimit system property.</p>	<p>10 (default)</p>

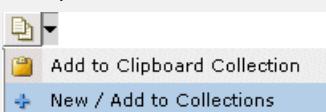
Update Function	Specifies a method name in the JPO given in the Update Program setting. See Update Program for more information.	The name of a function in the Update Program JPO, such as: setAssignedBuyerDesk setPackage Access setPartClassification
Update Program	Specifies a JPO that contains a method to set the field value when the field is displayed on an Edit mode form and when the user clicks Done. This program is used only when the Field Type setting is program or programHTMLOutput. Using an update program is recommended only when the Field Type is not attribute or basic. See JPO for Getting Field Values .	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
Validate	Specifies the method execute for validating any cell. This setting is deprecated. Existing forms that use this setting will still work, but new forms should use the OnChange setting instead.	The name of a method, such as: checkUniqueName
Validate Type	Specifies how field values should be validated. This setting is deprecated. Existing forms that use this setting will still work, but new forms should use the OnChange setting instead.	Basic or Restricted--The field values are validated against the value of emxFramework.Javascript.BadChars. Name--The field values are validated against the value of emxFramework.Javascript.NameBadChars.
Vertical Group Name	All fields with the same Vertical Group Name will display in a column in the specified form location.	<text name of group>
View Exclude	Comma-separated list of attributes that will not be included in the webform when opened in View mode when a field on the webform has been defined as Field Type = Dynamic Attributes.	For example: attribute_Cost
Window Height	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. Use to define the height of the popup chooser window or window opened from any href link in a form field.	Number of pixels, such as 700. The default is 600.
Window Width	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. Use to define the width of the popup chooser window or window opened from any href link in a form field.	Number of pixels, such as 700. The default is 600.

*Required Setting

URL Parameters Accepted by emxForm.jsp

This table lists the parameters available to emxForm.jsp. You can add these parameters to the href parameter for the component that calls the form. For example, when you specify the emxForm.jsp to be called from a tree category, you can add these parameters to the href parameter for the menu object.

Parameter	Description	Accepted Input Values
appendFields	The name of the web form that defines fields that you want to include in the form being defined.	
appendURL	Defines additional parameters to append to the URL string and is specified in this format: <code><KEY> SuiteKey</code> Where the <code>KEY</code> is a value in the properties file of the specified <code>SuiteKey</code> . See Additional URL Parameters for Forms .	Effectivity EngineeringCentral
displayCDMFileSummary	Determines if the file summary list should display on the same page as the Properties for a CDM object.	true false (default)
editLink	Use to have the system include a default Edit toolbar item in the View Form header.	true false (default) emxForm.jsp?form+ENCPart&editLink=true
*form	Specifies the web form administrative object used for presenting the form page. You must use the actual form name: symbolic names are not supported.	Name of web form administrative object. emxForm.jsp?form=ENCPart
formHeader	Use to define the content of the heading that appears at the top of the form page, in the header frame. The value can be either a string resource id or the label itself. The text in the label and string resource ID can contain any valid select expression.	emxForm.jsp?form=ENCPart&formHeader=Properties emxForm.jsp?form=ENCPart&formHeader=Properties:\$<type>\$<name> emxForm.jsp? form=ENCPart&formHeader=emxQuoteCentral.AssignedPackages.AssignedPackages
HelpMarker	Specifies the name of the help marker to call for context-sensitive help. The Help link always displays for form pages.	The naming convention for help markers is the page title, as displayed at the top of the visual page, prefixed with "emxhelp". The marker is all lowercase with no spaces.
jpoAppServerParamList	Allows session data to be passed to a JPO and uses the format: <code>scope:attributeName</code> where scope can be one of these values: <code>application</code> <code>session</code> <code>request</code> and the attribute must be a valid attribute used within the specified scope. The parameter can pass a comma-separated list of <code>scope:attributeName</code> values. The attribute values must be serializable.	application:<attributeName>,session:<attributeName>,request:<attributeName>
launched	Specifies the behavior of the popup page. You can change the behavior of the popup page from the normal mode pages using this parameter. When the user clicks the Launch button from a channel tab, <code>launched=true</code> is passed to the new popup page which indicates that the popup is a result of clicking the Launch button.	true false (default)

	In normal mode, pages will not have the launched parameter and will default to <code>launched=false</code> .	
mode	Specifies whether the form page should be view or edit mode.	view (default)--form is read only edit--form is editable <code>emxForm.jsp?form=ENCPart&mode=edit</code>
objectId	Use to specify the business object that the properties need to be displayed for.	Valid business object ID. <code>emxForm.jsp?objectId=3243.32424.232</code>
portalMode	<p>Every page configured inside the PowerView includes the parameter <code>portalMode=true</code>, so that the page can differentiate between normal display and portal display.</p> <p>When the Launch button is clicked, it launches the currently displayed channel tab into a maximized popup window, passing the parameters <code>launched=true</code> and <code>portalMode=false</code> to the new window.</p>	true false (default)
postProcessJPO	<p>Used to specify the post processing JPO program name and the method name to invoke after form processing and database update.</p> <p>This parameter is applicable only for edit mode.</p>	<JPO Name>:<Method Name> For example: <code>emxPart:processECO</code>
postProcessURL	<p>Specifies the name of the JSP executed during edit form post processing. The JSP is executed only after the form processing and database update complete.</p> <p>This parameter is applicable only for edit mode.</p>	<code> \${SUITE_DIR}/emxCustomPostProcess.jsp</code> <code> <AppDirectory>/emxCustomPostProcess.jsp</code> For example: <code>engineeringcentral/emxCustomPostProcess.jsp</code>
preProcessJavaScript	Defines a JavaScript function and is called after the form is loaded in edit mode to execute custom processing. Can be a single function, or a semi-colon separated list of functions.	JavaScript function name
PrinterFriendly	Specifies whether the form page should include the Printer Friendly tool.	true (default) false
RegisteredDirectory	Specifies the directory where the help files are located.	Directory name within ematrix.
relId	Specifies the relationship used to get the relationship attribute values for relationship fields.	Valid relationship ID. <code>emxForm.jsp?relId=3243.32424.232</code>
renderPDF	Specifies whether the Render PDF icon displays in the form toolbar in View mode. If the parameter is not passed in, the default is false and the icon is not shown.	true--The Render PDF icon displays in the form toolbar. false--The Render PDF icon is not displayed in the form toolbar.
showClipboard	<p>This menu is enabled by default and adds the  to the page toolbar. The icon acts as a pull-down menu:</p>  <p>If the user clicks , selected objects are added to the clipboard collection for that user. If the user clicks the arrow, the user can select the New/Add to Collections or</p>	true (default) false

	<p>Add to Clipboard Collection command.</p> <p>The showClipboard menu command only shows in the View form; the Edit form does not support the clipboard/collections functionality.</p> <p>Set the value for this parameter to false to disable this feature.</p>	
showPageURLIcon	<p>When true,  shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false,  does not show in the toolbar.</p> <p>The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in <code>emxSystem.properties</code>.</p>	true false
showTabHeader	<p>Applies only in Portal mode (when the page is displayed within a PowerView), enables or disables the page header.</p> <p>If false, the header text is not displayed in the PowerView tab.</p> <p>If true, the header text shows in the tab.</p>	true false (default)
submitAction	Determines the action to take after completing the form (user clicks Done or other action button other than Cancel, in addition to closing the form window):	refreshCaller: Reload the calling page. If called from a table or structure browser, then reloads the table or structure browser. doNothing: When called from a table or structure browser, does not refresh (including sorting) the calling page. treeContent: Load the newly-created object's tree in the main content frame. treePopup: Load the newly-created object's tree in a new pop-up window.
targetLocation	When a URL is defined to open in the slide-in frame (the <code>Target Location</code> setting for the component is set to <code>slidein</code>), BPS appends this URL parameter with this value. This parameter supports a custom page where the html for a popup window needs to be different than the html for a slide-in window.	slidein
TipPage	Specifies the Web page (html or jsp) to launch when the user clicks one of the page toolbar buttons in the form header frame.	Name of a custom URL page. <code>emxForm.jsp?form=ENCPart&TipPage=../myapplication/showMyTipPage.jsp?</code>
toolbar	<p>Specifies the menu administrative object used in the header frame that represents an action or filter toolbar.</p> <p>The value can be a comma-separated list of menus/toolbars.</p>	PMCWBSTaskToolbar ECEBOMToolbar,ECEBOMFilter

*Required

Actions Menu in a Configurable Form

You can only use the Actions menu in View mode of the form page. You add an Actions menu to a View form page by passing the menu name through the URL parameter to emxForm.jsp.

The following topics are discussed:

- [Example Actions Menu](#)
- [Using Default Edit Link](#)
- [Custom Page and Refresh](#)

Example Actions Menu

The following example URL shows how to pass the Actions menu for the View mode form. Because the default mode is View, the mode parameter does not have to be passed.

```
 ${COMMON_DIR}/emxForm.jsp?form=BuyerDesk&objectId=3243.32424.232  
&relId=3432.2342.2344&formHeader=BuyerDesk&toolbar=SCSBuyerdeskToolbar
```

You can assign the parameter actionMenuName to any valid menu object configured as an Actions menu. These menus contain the command objects that represent the toolbar items. This section describes some specific uses of Action menus with the configurable form. For instructions on configuring an Actions menu and toolbar items, see [Menus](#).

Using Default Edit Link

The View mode form page supports a default Edit link that displays when the parameter "editLink=true" is passed to emxForm.jsp. Here is an example of the default Edit link.

Compact Copy Machine: Properties	
Categories	
Details	
Name	Compact Copy Machine
Primary Image	
Type	Product Line
Description	Includes all Compact Copy Machines
State	Complete
Owner	DesignEngineer, Test
Originator	DesignEngineer, Test
Program	
Originated	1-Jan-2011, 7:01
Modified	Jan 31, 2011
Policy	Product Line
Vault	Service Documentation

When a user clicks this link, the same form page opens, with the same web form object name, but in Edit mode. Edit mode can also be configured to open in a slidein frame, or to convert the existing View mode into Edit mode.

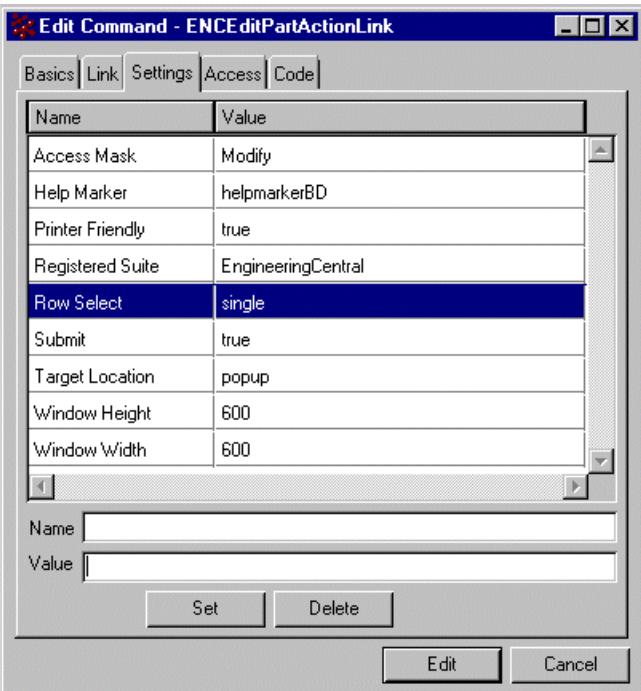
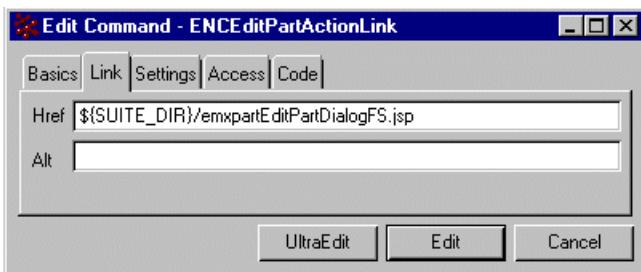
Custom Page and Refresh

You can configure an action menu link to edit the object displayed in the View mode form page. Upon successful update, it then refreshes the View mode form to reflect the changes.

To implement this kind of toolbar item, the action menu link command object must have these settings:

- Row Select=single
- Submit=true
- Target Location=popup

Here is an example of a command configured with these settings.



After editing the selected item, you can refresh the View mode form page by calling an appropriate JavaScript API method, such as:

```
//do Form refresh
top.opener.top.refreshViewFormPage();
top.close();
```

To perform a form refresh from a modal window, you must call the JavaScript method "releaseMouse()" to release the mouse. The following JavaScript method shows how to implement this.

```
function doDone() {
//check for page errors here, if there are no errors,
// Submit the data to edit/create the item
//then continue on to do the following:
top.opener.top.modalDialog.releaseMouse();
//do any page redirects/refreshes here - (Form refresh)
top.opener.top.refreshViewFormPage();
top.close();
}
```

If the Cancel button in a modal window refreshes the page underneath it, you must use the sample code shown above. If the Cancel button simply closes the window (top.close()), then it is not needed.

Filter Toolbar for a Configurable Form

You can define a custom filter toolbar that displays beneath the Actions toolbar in a configurable form. The filter toolbar only displays in View mode.

The filter toolbar is defined as a separate toolbar (see [Menus and Toolbars](#)), and is passed to emxForm.jsp as a URL parameter.

On a form that uses framesets, any toolbar that includes a drop-down menu must be the bottom toolbar. When passing the toolbars to the form, they show in the order in which they are passed so the toolbar with the menu must be the last toolbar listed.

This custom toolbar can use the standard toolbar commands, such as the AEFSeparator, plus the input controls defined in [Toolbars](#).

If you want to provide a default value for the filter, use the `Default` setting on the input control. When the form loads, the default values for the input controls are available in the requestMap to fields defined with a Field Type of `program` or `programHTMLOutput`. The command names are used as the parameter names in the requestMap. The default values are also available to a specified JPO using an embedded table using the `Field Type = emxTable`.

To apply the filtering, use `href= applyFormFilter()`, provided as part of the JavaScript API for the framework. The API posts the values entered in the filter input controls, plus the original set of input parameters, to emxForm.jsp. Based on these values, the form output can be changes for fields defined with a Field Type of `program` or `programHTMLOutput`. For an embedded table, the new values are retrieved from the requestMap.

In this example, the filter toolbar includes 2 input controls: one for the Plant Name and one for the submit button (labelled Filter).



The Plant Name control is defined with these settings:

- Input Type = combobox
- Range Values = Boston, New York, Chicago, Atlanta, Unassigned
- Label = Plant Name
- Default = Unassigned

The submit button is defined with these settings:

- Input Type = submit
- Action Label = Filter
- `href= applyFormFilter()`

When the form is loaded, the default value of Unassigned is used to populate the form. If the user selects a different value and clicks the Filter button, the form is updated with the values for the selected Plant Name.

Design Considerations for Form Fields

This section describes some of the decisions you should make before configuring form fields. It lists the options available for different kinds of fields, and lists the parameters and settings to implement the various options.

For descriptions of the parameters and settings referenced in this section, see [Parameters for Web Form Objects](#). For use case examples of the implementing features, see [Form Fields](#).

The following topics are discussed:

- [Design Questions](#)
- [Entering Field Values Using Type Ahead](#)
- [Using Type Ahead in Custom Forms](#)
- [Units of Measure in Custom Forms](#)
- [Entering URLs and mxLinks in Form Fields](#)
- [Using Dynamic URL Links in Custom Pages](#)

Design Questions

How should field values be obtained?

When designing form fields ask yourself these questions when deciding how to obtain values:

- Can the values be retrieved using a select expression applied to the current business object or relationship?
If yes, this is the preferred mechanism. Use the Expression and Applies To parameters for the field.
- Is the value an image file?
If yes, then use the Field Type=image and Image=FILENAME settings.
- Can the values be obtained using a JPO?
If the values cannot be obtained using an expression or image file, then a JPO can be used. Use the Field Type=program or Field Type=programHTMLOutput setting and the program and function settings to specify the program and function names.

Who should have access to the field?

Define the users, groups, and roles who can access the field. Form fields have the same access checks as the other UI components. See [User Access to UI Components](#).

Does the field need to be translated?

If so, the labels should be translated in the StringResource.properties file for the application as described in [Internationalizing Dynamic UI Components](#). If the field's value is the name of an administrative object, policy state names, vault name, or ranges for an attribute, use the Admin Type setting.

When the Allow Manual Edit setting is true, the Admin Type is ignored.

How should the field be displayed on a View mode form?

- Should it be hyperlinked?

If yes, then enter the URL to call when the field value is clicked in the href parameter. By default, the system passes the object ID for the current object to the called page.

- Should the hyperlink pass the object ID for an alternate object?
If yes, use the Alternate OID expression setting to specify the expression that gets the object(s) to pass.
- Where should the href page be displayed?

Use the Target Location setting to specify whether the page displays in a popup window, a slide-in frame, or in a specific frame of the current window. Use Popup Modal to control the modality of the popup window.

- Should it have a type icon next to it?

If yes and the icon should reflect the current business object's type, use the Show Type Icon=true setting.

If yes and the icon should reflect the type for an alternate business object, use the Show Alternate Icon=true and Alternate Type expression=EXPRESSION settings.

Should the field be editable?

If yes, add the Editable=true setting or do not specify the setting (it defaults to true). Then see the options listed below for editable fields.

If no, add the Editable=false setting.

If editable, what characteristics should the field have?

- What type of control should be displayed so users can enter or choose a value?

Use the Input Type setting to specify the type of control, such as textbox, combobox, radiobuttons, etc. The default is textbox.

- If the control is a text box, should it have a Browse (...) button that calls a range helper window?

If yes, use the RangeHref parameter to specify the page to appear in the window. This page can be a common chooser page included with the framework or a custom page. To define the height and width of the window, use the Range Helper Window Height and Range Helper Window Width settings.

- Should the editable field be required?

If yes, add the Required=true setting. The field's label will be red and italic.

- Should the field values be validated?

If yes and you want to use standard Javascript validation, use the format setting and specify the type of data, such as date, real, integer, text.

If yes and you want to use a custom function to validate the data, use the OnChange setting.

- Is the field value an attribute or basic?

If yes, use the Field Type=attribute or Field Type=basic setting so the system updates the database according to the type of value.



Entering Field Values Using Type Ahead

The Type Ahead feature stores text values entered in form text fields, then displays the most recent entries the next time that the user enters that field. They can then select from the list rather than needing to type the complete value or use a chooser.

Type ahead can be achieved using any of these methods:

- Custom JPO that retrieve the list of possible values for a field (described in this section).
- Advanced Search that uses emxFullSearch.jsp to retrieve possible values for a field. See [Advanced Search Used for Type Ahead](#).
- Pre-defined Choosers for Person, Organization, or Type fields. See [Pre-Defined Type Ahead Choosers](#).

If a field is defined with both a custom JPO and advanced search for Type Ahead, then the custom JPO is used.

When defining a field, you can:

- Enable or disable type ahead
- Override the number of values to save per context user, per field
- Define a JPO and method to retrieve values from the database (not those previously typed by the user) based on what the user starts typing
- If a JPO is used, override the number of characters a user must type before the JPO is triggered

The number of values stored for each user per field depends on the value for the emxFramework.TypeAhead.SavedValuesLimit property in the emxSystem.properties file. The default is 10 unique values and can be overridden at the field level using the TypeAhead Saved Values Limit setting. Duplicate values are discarded so that each of the values is unique. The values are stored in the order they are submitted so that the oldest item is removed when the list is full. Although stored in submitted order, the values are displayed in alphabetic order.

Set the emxFrameworkTypeAhead property in the emxSystem.properties file to true to enable Type Ahead for all fields on all forms. You can explicitly disable field settings by using the TypeAhead=false setting. List box, date, and multiline fields are not supported. If the system property is set to false, Type Ahead is disabled even if the field setting enables it.

To avoid conflicts with the browser's built-in auto-complete capability, the JPO must set the HTML autocomplete attribute to false.

Field values are stored in the Revision field of a workspace Cue object in an XML document. The naming format used for the Cue object is emx_form_formname. One cue object stores all values for the relevant fields on the specified form for the context user.

Cues apply to the current context, so a user only sees the values they enter. The cues are created in the inactive state.

Note: Cues are used by the Matrix Navigator and PowerWeb to provide visual signals for specific objects. If a type ahead cue is made active, it will have no affect. See the *MQL Guide* for complete details about cues.

When the user starts typing or double-clicks in the field, the page pops up the list of stored values. The user can:

- Continue to type to shorten the list until 1 remains.
- Use the cursor to click on the needed value.
- Use the up/down arrow keys to highlight a value, then press Tab or Enter, or click on the value.
- Press the Escape key to close the popup.

If the user types a value that does not match anything in the list, the popup closes.

If the field uses a JPO:method, the functionality is slightly different. The field pops up the list of recently-entered values as described above. When the user types something that does not match any of those values and exceeds the character count setting, the JPO is triggered to return the list of matching values. This process results in the list first showing several values, the number of values gets shorter as the user types, and then the list may get longer when the JPO method returns matching

values.

To implement type ahead functionality, you need to define these settings on the field:

- `TypeAhead=true|false`. Enables type ahead. If not provided, automatically set to true. Typically used to disable type ahead for a field.
- `TypeAhead Program`. Name of the JPO called to retrieve a list of possible values. This JPO is called after the user types the specified number of characters. A JPO is typically used by fields that are normally populated using choosers.
- `TypeAhead Function`. The name of the JPO method used in conjunction with the `TypeAhead Program` setting.
- `TypeAhead Character Count`. Overrides the `emxFramework.TypeAhead.RunProgram.CharacterCount` system property. This setting may be useful for fields whose values may all contain the same prefix.
- `TypeAhead Saved Values Limit`. Overrides the `emxFramework.TypeAhead.SavedValuesLimit` system property.

You must also create the JPO and method called by the TypeAhead Program and TypeAhead Function settings.



Using Type Ahead in Custom Forms

If you want to implement the Type Ahead feature in a custom form, you can use the framework's tag library.

Use the `typeahead:saveFieldValues` tag in a processing page to save the field values. Use the `typeahead:displayFieldValues` tag to show the auto-complete behavior as described in [Entering Field Values Using Type Ahead](#). You must also create the JPO and method that the tag calls. The format for the save tag is:

```
<emxUtil:saveTypeAheadValues
    context="<% context %>"
    form="form1"
    field="field1"
    displayFieldValue="<% tagDisplayValue %>"
    hiddenFieldValue="<% tagHiddenValue %>"
    count="50" />
<emxUtil:commitTypeAheadValues context="<% context %>" />
```

Replace the italic text with the values for your custom form. The result of the display tag is a javascript array of the field values. It will also generate the javascript code that calls back to the server to execute the JPO if there is one. The format for the display tag is:

```
<emxUtil:displayTypeAheadValues
    context="<% context %>"
    form="form1"
    field="field1"
    displayField="field1Display"
    hiddenField="field1"
    program="emxTypeField"
    function="genValues" />
```



Units of Measure in Custom Forms

If you want to implement the units of measure (dimension) in a custom form, you can use the framework's tag library. See [Units of Meaure for Columns](#) for details on using the UOMAttribute tag.



Entering URLs and mxLinks in Form Fields

When a field has the Dynamic URL setting set to enable, users can enter URLs or mxLinks in a create or edit form, and the system displays those values as hyperlinks in the view form. The valid URL formats are defined by the `emxFramework.DynamicURL.Keywords` parameter in the `emxSystem.properties` file.

The `emxFramework.DynamicURLEnabled` system parameter controls this feature. If enabled at the system level (in the `emxSystem.properties` file), it can be disabled for specific text fields by passing the `Dynamic URL = disable` setting (the default value is `enable`).

Dynamic URLs can be entered in any free-form text fields. Dynamic URLs *cannot* be entered in a field if:

- The Dynamic URL setting is set to disable (entered URL value will display as text)
- The Field Type setting is set to programHTMLOutput
- The Format setting is set to date or numeric
- Configured with businessobject or relationship expressions as attributes with datatypes Boolean, date/time, real, integer or associated with a dimension
- Configured with businessobject or relationship expressions as attributes with predefined fixed range values
- Configured with businessobject basic expressions as attributes (such as Type, Name, Rev, Vault, Owner) except Description

Users enter the URL in straight text, such as "`http://server.com\abc.jsp?param=value`" or `www.enovia.com`, and then the system converts the text to hyperlinks.

When Dynamic URLs are enabled, mxLink values are also enabled. Users can enter an mxLink value in a text box, and the framework converts that entry into a hyperlink. The user can type in the mxLink keyword and value, or search for the mxlink value. See the *Application Exchange Framework User's Guide* for details on how a user enters the value.

The format entered in the text field is:

```
mxlink:<type>|<name>|<revision>|<vault>
```

For example:

```
mxLink:Part|Part-5000-01|0|eService Production
```

This string is translated to a URL pointing to the default page for the object, such as:

```
http://server:8080/ematrix/common/  
emxNavigator.jsp?objectId=32902.4369.56921.33541
```

and displays as a hyperlink such as:

```
Part Part-5000-01 0
```

When processing the `<type>` of an mxLink, Business Process Services uses the symbolic name (for example, `type_Part`), and looks up that symbolic name in the string resources properties file for the language the browser is set to and displays the internationalized name of the type. If the string resources properties file does not contain an entry for the type's symbolic name, then the type displays as entered.

See the *Application Exchange Framework User's Guide* for variations on this format, such as omitting vault or revision, and using the latest revision identifier. If a user omits the vault or in the mxLink, the framework automatically adds the vault to the stored hyperlink. If a user omits the revision identifier, the framework automatically adds the latest revision identifier to the stored hyperlink.

On an edit page, if the field containing a dynamic URL or mxLink is defined as non-editable, then the field displays it as a link the same as on the view page.

When the user clicks Done on a page where an mxLink has been entered, the form validates the mxLink entries and displays an error message if any of the values could not be resolved. The user has the option of saving the form anyway, or canceling and fixing the error. If more than 10 errors are found, the message lists up to 10 errors and indicates there are additional errors.



Using Dynamic URL Links in Custom Pages

You can use the FormatURL taglib in any custom page or component. This taglib parses the provided text string and formats it as a hyperlink. This taglib can be used in freeform text fields.

The TagLib uses these parameters to configure the displayed converted data.

TagLib Parameter	Description	Possible/Default Values
Text	URL strings to be parsed and replaced with the HREF anchor tags.	<code>www.3ds.com</code> <code>"http://server40:8080/ ematrix/common/ emxNavigator.jsp?suiteKey=Components& objectId=32902.4369.56921.33541"</code> <code>mxLink:Part P-5000-01 0</code>
externalLink	Determines which method to use to open a popup window. If set to yes, <code>window.open()</code> is used; if set to no, <code>showNonModalDialog</code> is used. If the hyperlinked data is visible within the ENOVIA product, use No. If the hyperlinked data is visible in an email box (such as when using the <code>mailto:</code> keyword), use Yes.	no (default) yes

The syntax for this taglib is:

```
<framework:FormatURL  
text=<text to be formatted>"  
externalLink=<Yes or No>"  
/>
```

See the following sections for more specific examples of this taglib's usage.

External URLs

When the text provided in the taglib code starts with any of the following strings, it will be formatted as a hyperlink:

- `http://`

[https://](https://www.ftp://)
• www.
• ftp.

The end of the URL is defined by a blank space.

If the URL is enclosed within double quotes, it can contain blank spaces. For example, this text string:

"<http://myserver/abc.jsp?param1=value1¶m2=My Value2>"

will be interpreted as

"<http://myserver/abc.jsp?param1=value1¶m2=My%20Value2>"

URLs are not case-sensitive.

URLs in Custom JSP Pages

For custom JSP pages, you can apply the dynamic URL feature to a description field. For example:

```
<tr><td class="label"><emxUtil:i18n localize="i18nId">  
emxFramework.Basic.Description</emxUtil:i18n></td>  
<td class="field"><framework:FormatURL text="<%>sDescAttr%>"/></td>  
</tr>
```

If the link opens an ENOVIA product page, the externalLink attribute is optional; if the link opens a page in an external application (such as an email application), then you should set `externalLink = Yes`.

Object Links Using mxLink Keyword

The text of the FormatURL taglib can use the mxLink keyword with the name/type/revision information to define the link.

The syntax for this keyword is:

```
<framework:FormatURL  
text=mxLink:<type>|<name>|<revision>  
/>
```

You can use spaces between the colon and pipe characters. The revision is optional. If not provided, the taglib uses the latest revision of the object. If the type/name/revision is not unique because of vaulting setups, the first found object is selected.

For example, this code:

```
<framework:FormatURL  
text=mxLink:Part|P-5000-01|0  
externalLink="No"  
/>
```

will show this as the label:

Part P-5001-01 0

with this as the HREF:

"<http://server40:8080/ematrix/common/emxNavigator.jsp?suiteKey=Components&objectId=32902.4369.56921.33541>"

You can also use the ObjectId with the mxLink keyword. This code produces the same results as shown above when providing the type/name/revision:

```
<framework:FormatURL  
text=mxLink:32902.4369.56921.33541  
externalLink="No"  
/>
```

If the Type or Name includes the pipe character, you must use a double-pipe as the separator character. For example, `mxLink:Part||Mfg Part|001||0` where Mfg Part|001 is the object name. You cannot mix single and double pipes as separators in the same string.

Form Fields

This section describes how to use the parameters and settings for a web form administrative object to define the form's fields. If there is more than one value for a field, the form displays all the values when in View mode. In Edit mode with the setting `Editable=true`, the field only displays the first value.

In this section:

- ❑ [Field Values as Select Expressions](#)
- ❑ [Field Values Obtained from a Program](#)
- ❑ [Field Values as a Hyperlink and Type Icon](#)
- ❑ [Field Values with Hyperlinked Data Using an Alternate OID and Alternate Type Icon](#)
- ❑ [Field Values with an Additional Icon](#)
- ❑ [Field Values as Hyperlinked Image](#)
- ❑ [Field for Attribute with Choices in Combo Box](#)
- ❑ [Field for Attribute with Choices in Radio Buttons](#)
- ❑ [Fields with Checkboxes](#)
- ❑ [Field for Attribute with Choices as Checkboxes](#)
- ❑ [Field Value with Dates](#)
- ❑ [Field Values as Units of Measure](#)
- ❑ [Form Fields as Arithmetic Expression](#)
- ❑ [Fields as Dynamic Textareas](#)
- ❑ [Field as Section Header and Separator](#)
- ❑ [Fields that are Grouped](#)
- ❑ [Fields Arranged in a Table](#)
- ❑ [Field that Embeds a Configurable Table](#)
- ❑ [Field with Popup Range Helper](#)
- ❑ [Fields with Dynamic Attributes](#)

Field Values as Select Expressions

One way to get field data is based on a select expression applied to a business object or relationship. The select expression might return one or more values.

For example, to get data for a Current State field, you can use a select expression applied to the current business object. To get the value for the Find Number attribute, which is an attribute on a relationship, you can use a select expression applied to the relationship whose ID is passed to the form page. This graphic shows examples of these two fields on a form in View mode.

Current State	Release
Find Number	55

The fields would look like this when the web form administrative object is displayed on a form page in Edit mode.

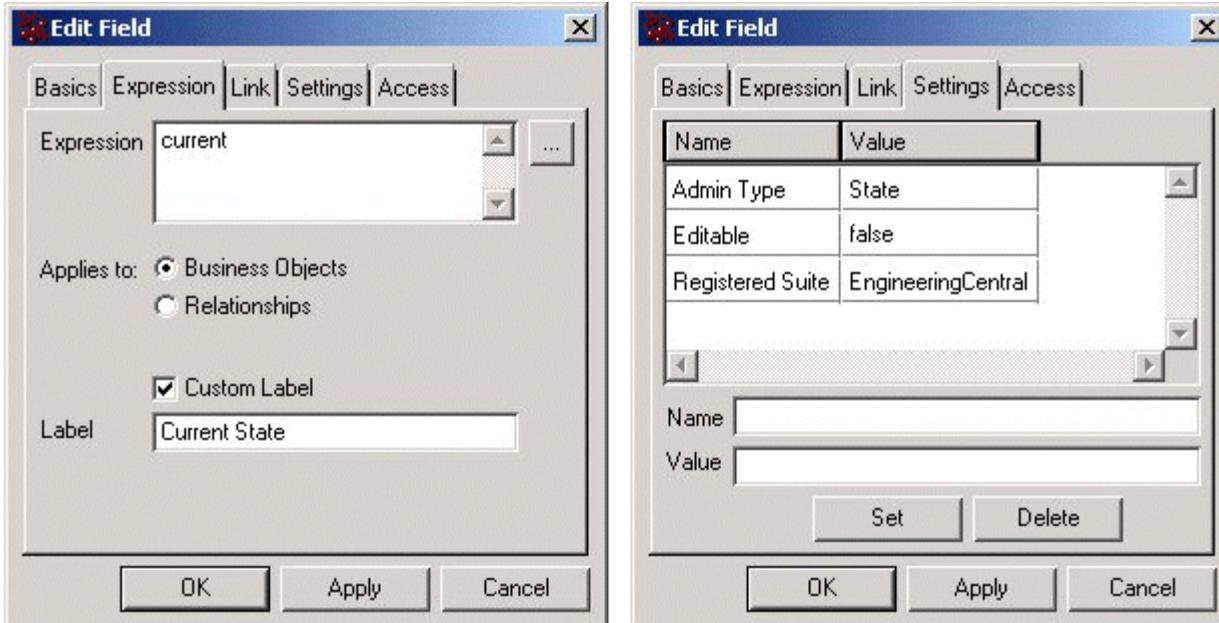
Current State	<input type="text" value="Release"/>
Find Number	<input type="text" value="55"/>

To configure a field so the data is obtained from a select expression, use these parameters and settings:

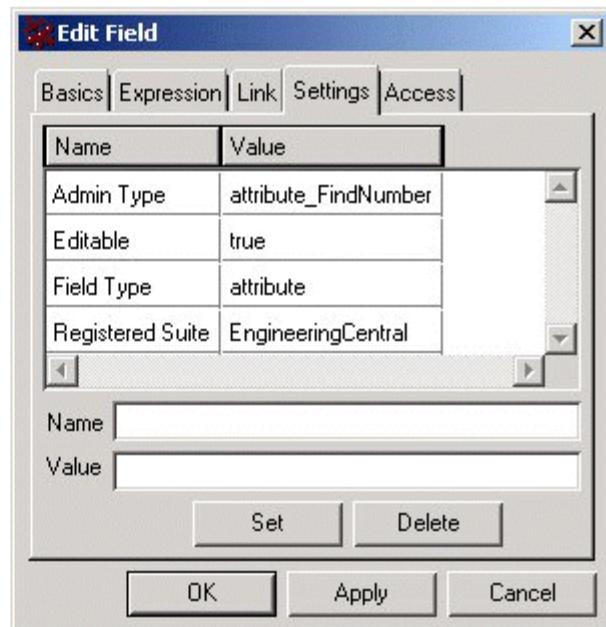
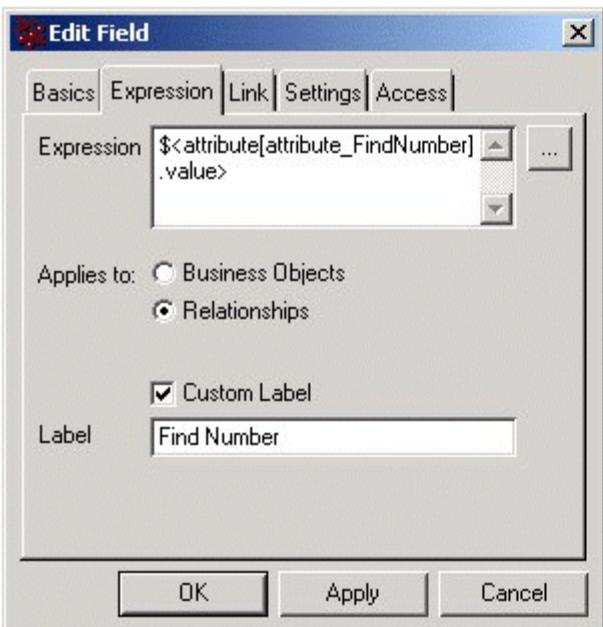
- Expression parameter--Enter the select expression to apply to a business object or relationship to get the field values.
- Applies to--Choose what the expression should be applied to: Business Object or Relationship.
- Field Type setting--When the expression is a business object basic or an attribute and the field is editable, you must include the Field Type setting. This lets the system correctly update the basic or attribute property.

The graphics below show how the Current State and Find Number fields shown previously are configured in the Business Modeler. In addition to the parameters that get the data using a select expression, the fields have settings that define the application associated with the field and that specify the administrative type so the value can be translated.

These dialog boxes show how to configure a field to get data from an expression applied to the current business object.



These dialog boxes show how to configure a field to get data from an expression applied to the current relationship.



Field Values Obtained from a Program

Field values can also be obtained from a method in a JPO program. The method is defined to obtain the values based on specific business logic.

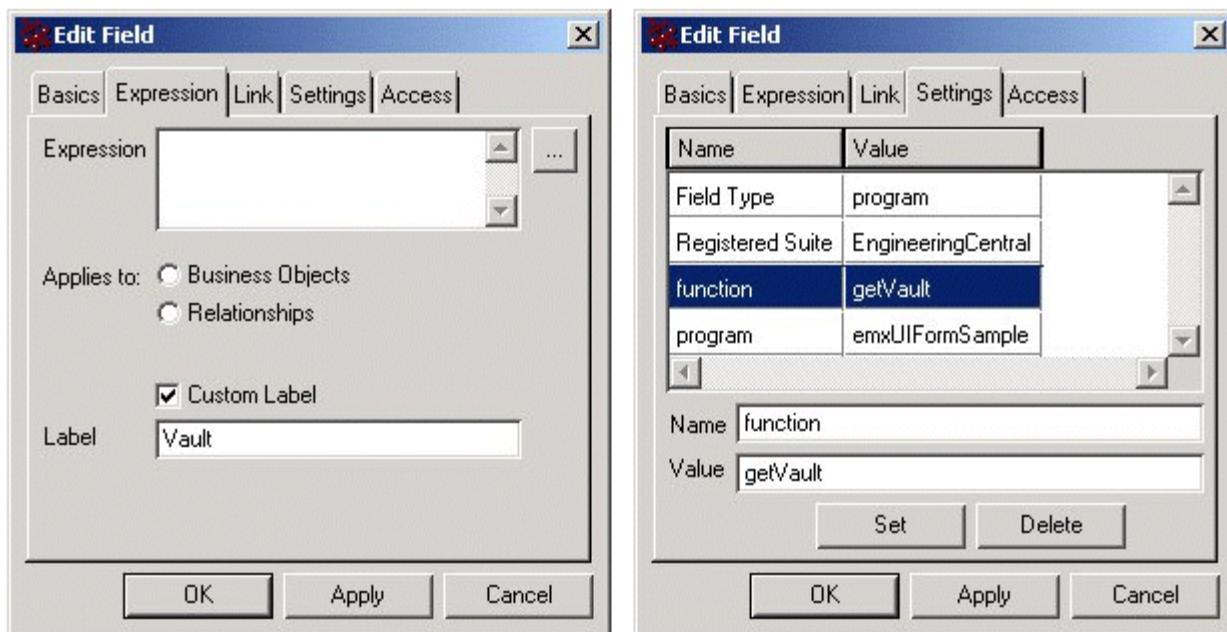
To get field values using a method in a JPO, use these parameters and settings:

- Expression parameter--Leave blank.
- Field Type setting--Set to `program` when program results contain only field values. Set to `programHTMLOutput` when program results contain complete HTML tag (including the field value) to be displayed in the form row.
- Program setting--Enter the name of the JPO object to be used for getting the field values.
- Function setting--Enter the name of the method in the JPO that should be implemented to return the field values.

To see a sample JPO that includes a method to get field values, see [Sample JPO for Getting Field Values](#).

The next graphics show how a Vault field could be configured in Business Modeler to get data from a JPO.

For simplicity, these example screen shots show a field that gets the current vault using a JPO program, even though the vault can be obtained more directly using the select expression "vault". The dialog boxes show how to configure a field to get data from a JPO program.



Field Values as a Hyperlink and Type Icon

Fields can be configured to display data as a hyperlink and with a type icon to the left of the hyperlinked text data.

For example, suppose you want a field to display the name of the current business object with the object's type icon displayed to the left of the name. In View mode, the name is hyperlinked. When a user clicks the link, the system passes to the href URL the ID for the business object the page applies to. The ID is passed using the objectId parameter. The View mode for such a field is shown below.



When this field is displayed on a form in Edit mode, the data is not hyperlinked because Edit mode doesn't display hyperlinks. The field would look like this.



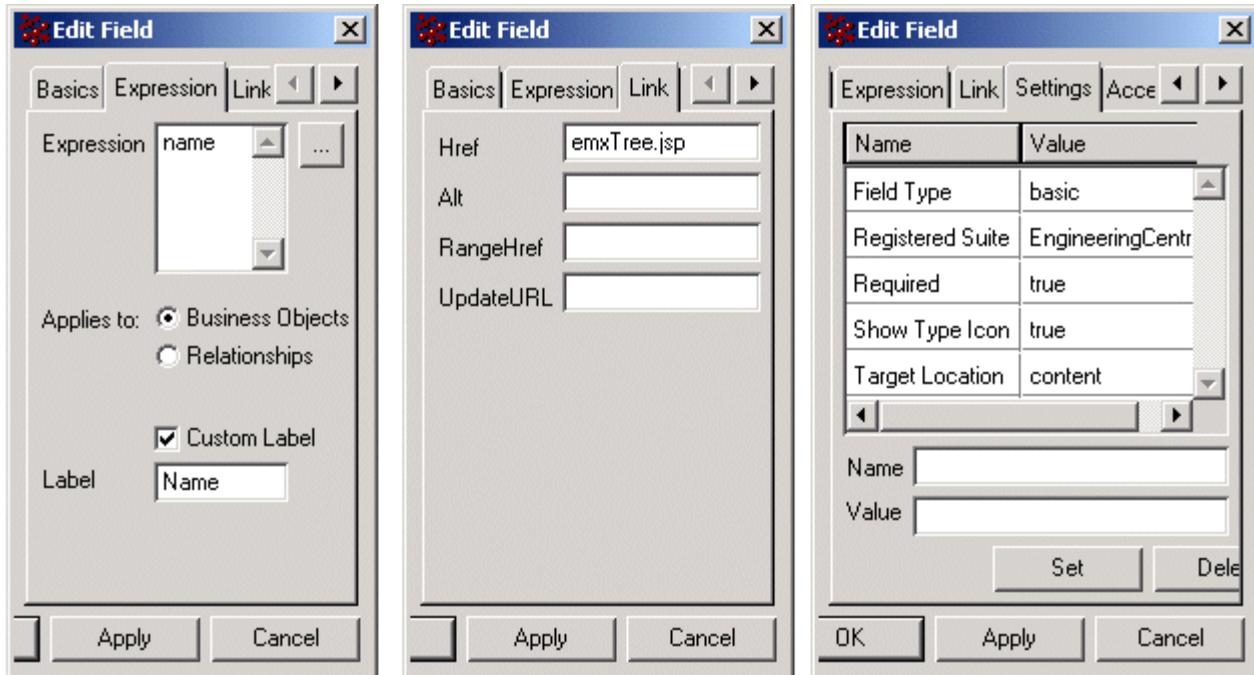
To configure a field so the data is hyperlinked when shown on a form in View mode, use these parameters and settings

- href parameter--Enter the JSP to call when the user clicks the hyperlinked data. The JSP can include parameters accepted by the JSP and path information, including directory macros.
- Target Location setting--Use this setting to specify where the href JSP should be displayed: popup, content, _top, or any valid frame name.

To configure a field to include the type icon for the current object, use this setting

- Show Type Icon setting--Set this setting to true.

The following graphics show how the Name field, shown previously, is configured in Business Modeler. In addition to the parameters and settings that make the data hyperlinked and that add the type icon, the field has settings that make the field required when the form is in Edit mode.

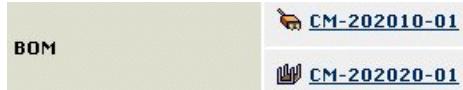


Field Values with Hyperlinked Data Using an Alternate OID and Alternate Type Icon

When a user clicks a hyperlinked field, the field passes an object ID to the href URL using the objectId parameter. By default, the ID field is for the object the page applies to. Another way to configure a hyperlinked field is to have the field pass the ID for a different business object.

The alternate object IDs are obtained from the expression assigned to the setting Alternate OID expression. The href parameter and therefore this setting are applicable only for the View mode form and the Edit mode form ignores it. You can also configure fields to show a type icon other than the type icon for the object the form page applies to.

This graphic shows a field in View mode that is configured to show hyperlinked data with an alternate OID and an alternate type icon.



This graphic shows the same field on a form in Edit mode.



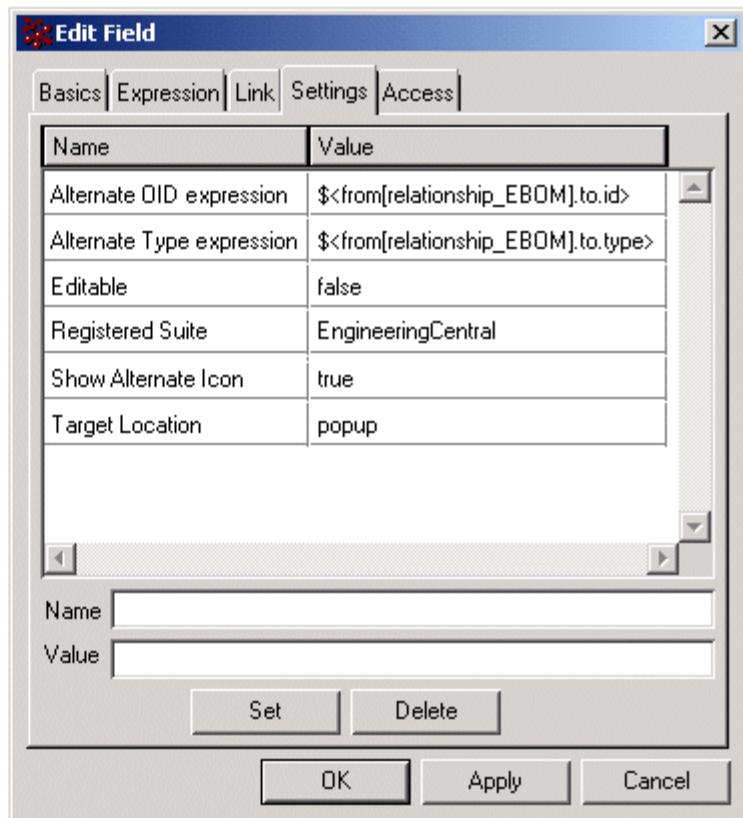
To configure a hyperlinked field to pass an alternate object ID, use these settings and parameters

- href parameter--Enter the JSP that should be called when the user clicks the hyperlinked data. The JSP can include parameters accepted by the JSP and path information, including directory macros.
- Alternative OID expression--Enter a valid select expression that returns one or more business objects. The system passes the IDs for the objects in the href using the objectId parameter.
- Target Location setting--Use this setting to specify where the href JSP should be displayed: popup, content, _top, or any valid frame name.

To configure a field to display a type icon for an object different from the object the form applies to, use these parameters and settings

- Show Alternate Icon--Set to true.
- Alternate Type expression--Enter a valid select expression that returns a business object. The system displays the type icon associated with this business object.

The following graphics show how the BOM field shown previously is configured in Business Modeler. The Link tab, not shown, would have emxTree.jsp for the href parameter.



Field Values with an Additional Icon

A field can include an icon in addition to the object name.

To do so, you can use any of these methods:

- Icon based on the object type (set using the emxFramework.smallIcon.defaultType property defined in the *Live Collaboration Administrator's Guide*)
- Alternate icon based on a select expression (see [Field Values with Hyperlinked Data Using an Alternate OID and Alternate Type Icon](#))
- JPO function to add an additional icon to the field based on the object's lifecycle state, or any other business requirement

The emxFramework.UIComponents.OverrideTypeIcon.Program=<JPO Name>:<Method Name> property in emxSystem.properties defines a global JPO:method used for all fields and columns that define the **Show Alternate Icon** or the **Show Type Icon** setting, and for all navigation trees. To use a different JPO:method for a specific form field, use the **Type Icon Function** and **Type Icon Program** settings.

The JPO:method specified in the emxSystem.properties file and the one specified using the **Type Icon Function** and **Type Icon Program** settings determine the icon to show in addition to the type or alternate icon already defined. For this JPO to be called, either the **Show Alternate Icon** or the **Show Type Icon** setting must be set to true. If both of those settings are set to false, the default JPO and any **Type Icon Function** and **Type Icon Program** values are ignored.

You can only use the additional icon if the value for the Field Type setting is any of these values:

- basic
- attribute

The function that retrieves the list of custom icons must return an image name or path for the id that is passed to it. The path, including the image name, must be specified relative to the ematrix\common directory. The image must be a gif image, and cannot include any spaces or special characters.

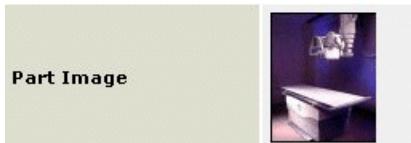
For details about the JPO method signature, input arguments, returned values and sample JPO code, see [Column Values Including Additional Icons](#).

Field Values as Hyperlinked Image

You can configure a field to show the primary image associated with an object.

See the *Live Collaboration Administrator's Guide* for details on the system properties that control default values for images in configurable components.

You can also configure a field to display an image only and this image can be hyperlinked when displayed on a form in View mode. This graphic shows an example of this type of field.



3D images show a 3D cue.

To configure a field as an image that links to the Image Manager or Image Viewer, use these settings

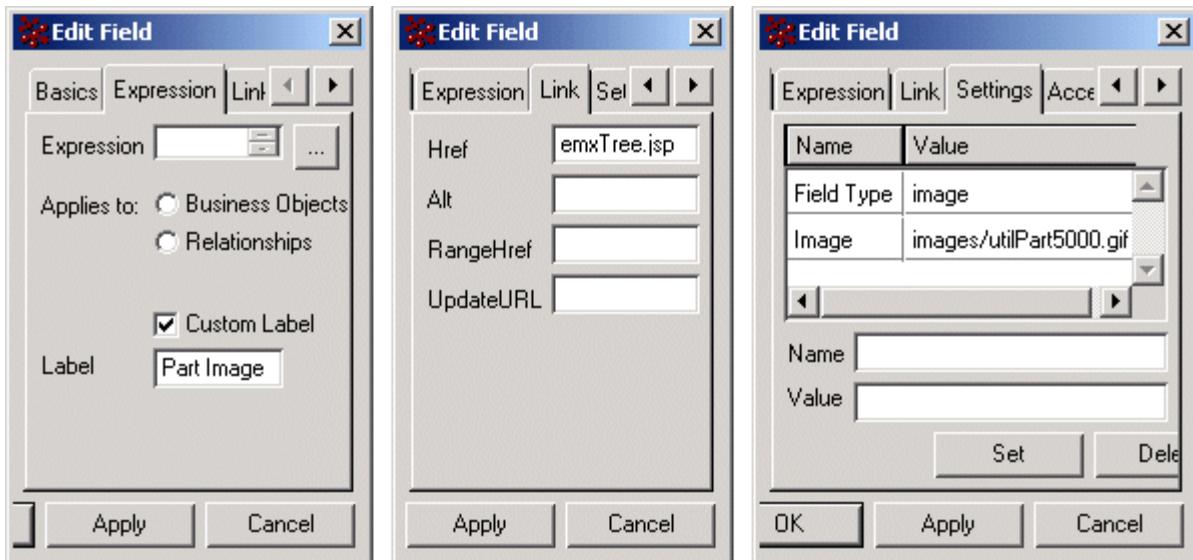
- Field Type setting--Set to image.
- Image Size setting--Set to format_mxSmallImage (default), format_mxMediumImage, format_mxLargeImage, or format_mxThumbnailImage.

You do not need to specify the image file name; the system uses the primary image associated with the context object. When a user clicks the image, the system opens the Image Manager or Image Viewer page, depending on the context user's roles.

To configure a field as an image that is hyperlinked in View mode, use these parameters and settings

- Field Type setting: Set to image.
- Image Size setting: Set to format_mxSmallImage (default), format_mxMediumImage, format_mxLargeImage, or format_mxThumbnailImage.
- Image setting: Specify the image file name. This file must exist in the application server (not in the database).
- href parameter: Enter the JSP to call when the user clicks the hyperlinked image. The JSP can include parameters accepted by the JSP and path information, including directory macros.
- Target Location setting: Use this setting to specify where the href JSP should be displayed: popup, content, _top, or any valid frame name.

The next graphics show how the Part Image field shown previously is configured in Business Modeler as a hyperlinked image.



Field for Attribute with Choices in Combo Box

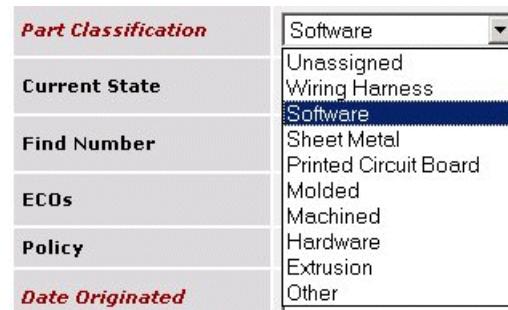
You can configure a form field to display an attribute (for a business object or relationship) that has choices (ranges).

Users can select one of the available choices using a combo box when the field is shown on a form in Edit mode.

For example, this graphic shows a field for a Part Classification attribute configured to show the choices in a combo box.



This graphic shows the same attribute with the combo box expanded to show all the ranges. The range Software is currently selected.



When in View mode, the currently selected attribute value is shown as read only text, as with this Part Classification attribute.

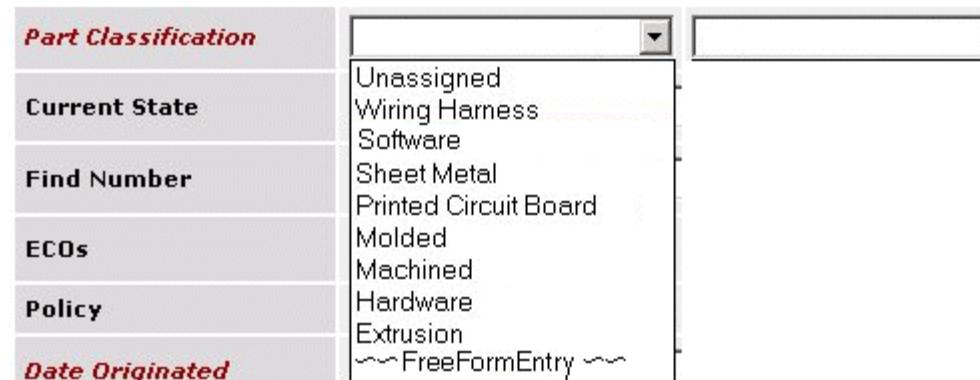


There are two ways to configure this kind of field depending on how you get the choices to populate the combo box. You can get choices to populate the combo box from ranges for the attribute in Business Modeler (for example, the Part Classification attribute), or from a JPO program.

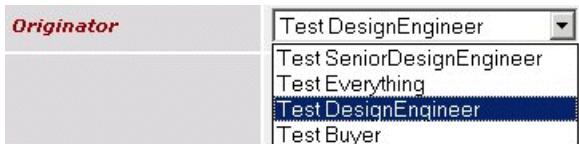
To configure a field to display attribute ranges in a combo box when the ranges are obtained from the attribute administrative object, use these settings

- Input Type setting--Set to combobox.
- Field Type setting--Set to attribute. This tells the system to obtain the ranges from the administrative object.
- Editable setting--Set to true (although it is true by default so you don't have to add it explicitly).

The below graphics show how the Part Classification field shown previously is configured in Business Modeler. It shows a field configured to display attribute ranges in a combobox, with ranges obtained from an attribute admin object.



For some attributes, it may not be practical to define the ranges in the administrative object, as with the Originator field shown below. In this case, configure the field to obtain the data from a JPO.

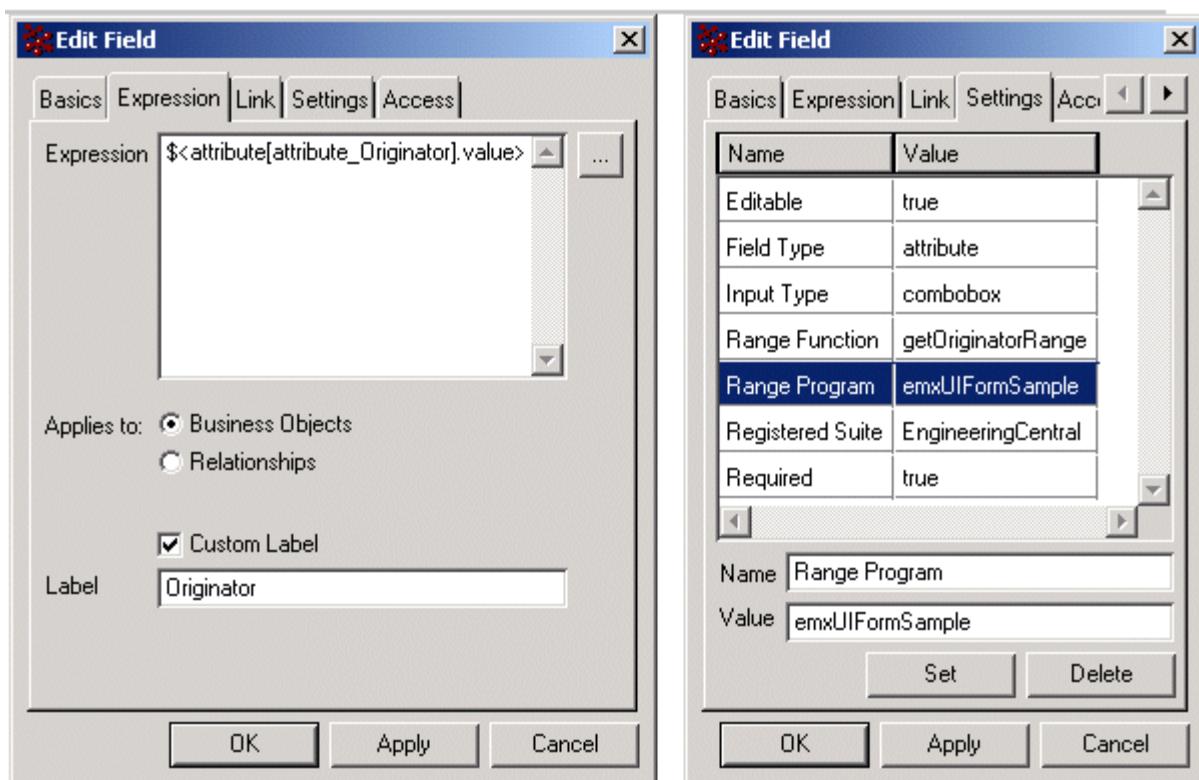


To configure a field to display attribute ranges in a combo box when the ranges are obtained from a program, use these settings:

- Input Type setting--Set to combobox.
- Range Program setting--Enter the JPO name containing the method to get the range values.
- Range Function setting--Enter the JPO method name to get the range values. The method should return the choices in the object type StringList.

For details of how to write the JPO for fetching the field choices, see [JPO for Getting Field Range Values \(Choices\)](#).

The below graphics show how the Originator field shown previously is configured in Business Modeler. The dialog boxes show how to configure a field to display attribute ranges in a combobox, with ranges obtained from a program.



Field for Attribute with Choices in Radio Buttons

You can configure a field to display an attribute as a radio button control.

The radio button controls are only shown when the field is displayed on the form in Edit mode. This example shows the Unit of Measure field configured with radio buttons.



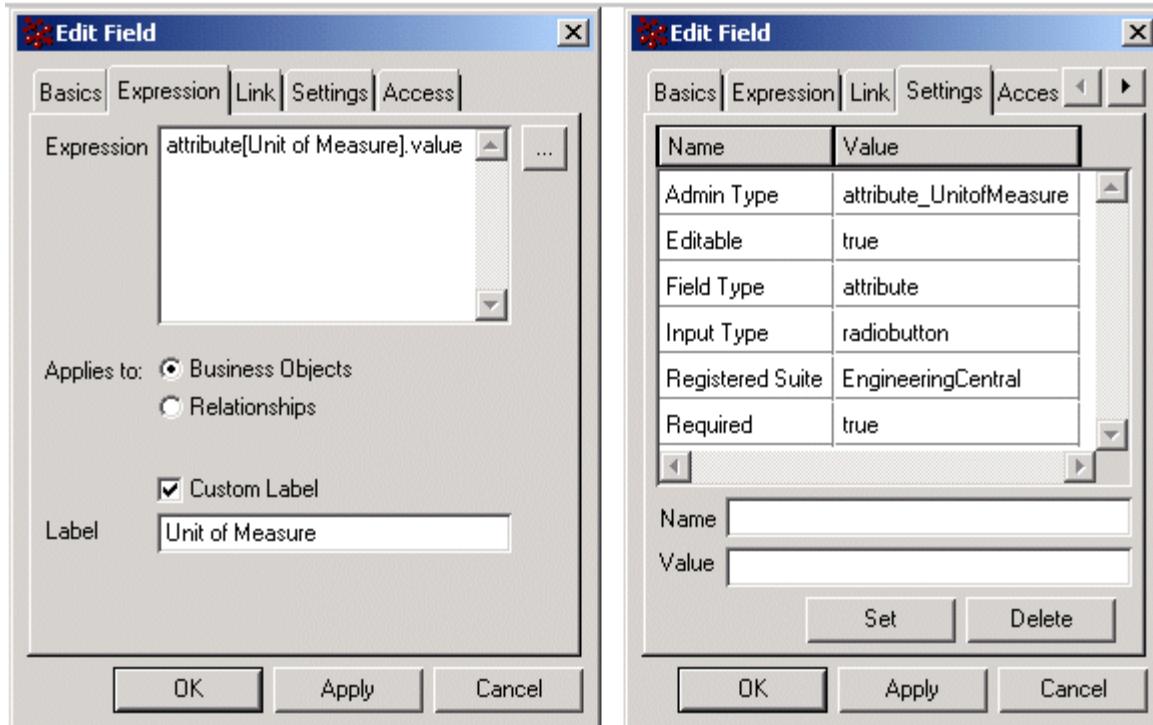
When the field is displayed in View mode, only the currently-selected choice is displayed.



To configure a field to show attribute ranges as radio buttons, use these settings:

- Input Type setting--Set to radiobutton.
- Field Type setting--Set to attribute. This setting and value causes the system to get the values from the database so the choices must be defined as ranges for the attribute in Business Modeler.

The below graphics show how the Originator field shown previously is configured in Business Modeler. The dialog boxes show how to configure a field to display attribute range as radio buttons, with ranges obtained from a program.



Fields with Checkboxes

If an attribute is defined with a datatype of Boolean, the form shows that attribute's label followed by a single checkbox in Edit mode. Checked=true; clear=false. In View mode, the value of the field (TRUE or FALSE) is shown.

The `Input Type = checkbox` setting does not need to be specified. You can specify the `Read Only Checkbox=true` setting on the field so it shows as a checkbox in View mode. As an alternative, you can use the `radiobutton` or `combobox` value for the `Input Type` setting to use those formats instead of a checkbox.

In addition, if the attribute datatype is defined as a string, it can also be treated as a Boolean if the range values for the string are listed in the `emxFramework.UIForm.Checkbox.BooleanValues` property in `emxSystem.properties`. The attribute must only have 2 values, and both values must match a pair defined for this property (not case sensitive). The field must also have the `Input Type = checkbox` setting defined (by default, range values show in a combobox).

The `Read Only Checkbox=true` setting is also required if you want to show the checkbox instead of the text value in View mode. For example, if the attribute datatype is string with range values of Yes and No, the form shows that field as a checkbox, where checked=Yes and clear=No (as long as Yes/No are defined in the property listed above).

Field for Attribute with Choices as Checkboxes

You can configure a field to display possible values for an attribute as a series of checkboxes.

The checkboxes are only shown when the field is displayed on the form in Edit mode. In View mode, the current value shows.

You can use a Range Program to specify the name of a JPO that contains a method to get the field value ranges (choices).

If you want to save multiple selections, a custom JPO needs to be written using the checkbox or listbox input types which might delimit the choices in the attribute using the Studio Customization Toolkit.

Field Value with Dates

You can configure fields to display the field value in date format.

The value must be a valid date string. The date value is displayed based on the system and browser setting using the IzDate taglib. Here is an example of a date field in View mode.

Date Originated Jul 9, 2002 10:04:40 PM

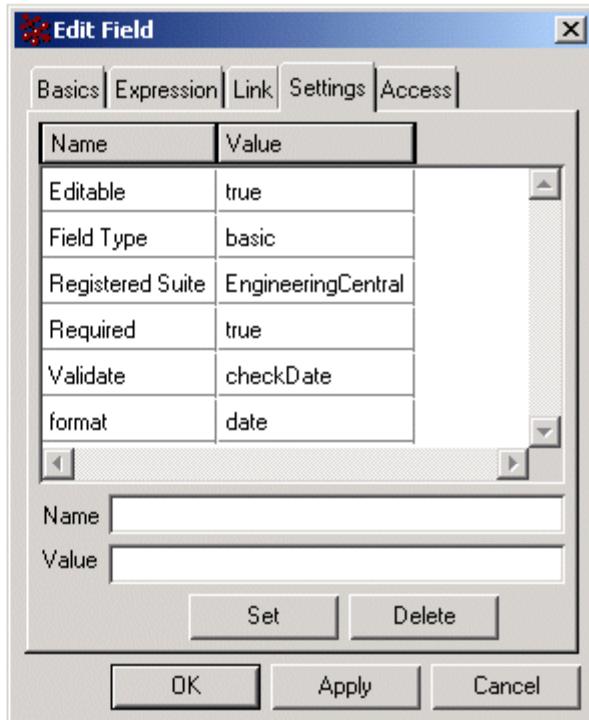
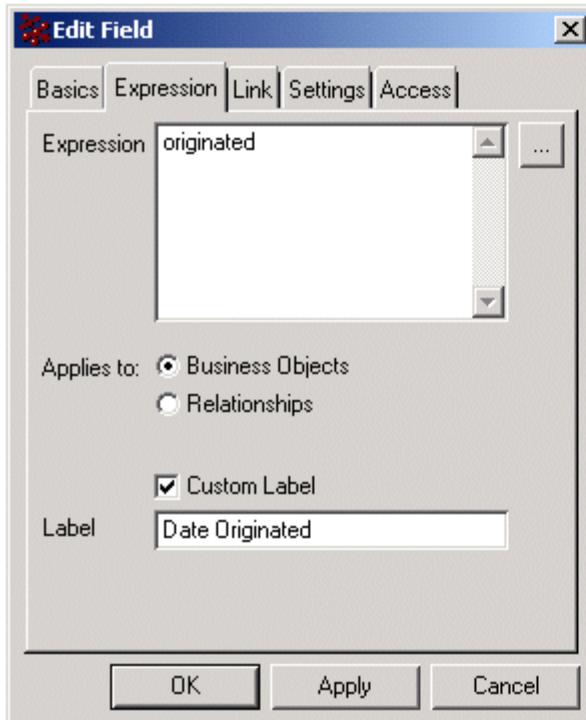
This is the same date field in Edit mode. The field includes the Calendar icon that users click to select a date from a Calendar popup.

Date Originated Jul 9, 2002 10:04:40 PM 

To configure a field to display a date and allow date input with the Calendar popup, use these settings

- format setting--Set to date
- Editable setting--Set to true

These dialog boxes show how to configure a field to display a date.



Field Values as Units of Measure

For attributes configured with a dimension, the units of measure can be displayed in any of the units defined for the dimension.

See the *Business Modeler Guide* for instructions on defining dimensions.

Form Fields as Arithmetic Expression

You can include the Arithmetic Expression setting for a form field so that the data in that field is the result of calculations using values from other form fields.

The expression can use any arithmetic operations (+ - * / %), arithmetic functions, or aggregation functions. The arithmetic functions include all those supported by the JavaScript Math class.

As an example, say you want to calculate the volume on a form that includes the Height, Width, and Length fields. You could perform this calculation using this expression:

```
"Height * Width * Length"
```

Since this expression is defined as a setting on the Volume field, the result displays as the value for that field.

Fields as Dynamic Textareas

When the Input Type = dynamictextarea setting is defined, a user can enter multiple values in that field separated by newline characters (the Enter key).

A blue triangle in the upper right corner indicates a field defined with this input type. The default size of a dynamic text area field is 20 (pixels); you can use the Field Size setting to change this value. See [Settings for Fields in Web Form Objects](#).

The screenshot shows the ENOVIA Search: General interface. At the top, there are buttons for Actions, Search Types, Reset Criteria, and a dropdown menu. Below these are two search criteria fields: 'Type' set to 'ECO' and 'Name' which is currently empty. A blue triangle in the top right corner of the 'Name' field indicates it is a dynamic textarea.

A user can click in the box, and a textarea opens where any number of values can be entered.

The screenshot shows a portion of a form interface. On the left, there are two columns: 'Name' and 'Revision'. The 'Name' column contains a dynamic textarea. The text area is currently empty but has a cursor at the top-left corner, indicating it is active for input. It features scroll bars on the right and bottom.

The user must tab or click outside of this textarea to finish entering the data for the field. The values are shown as a single string with values separated by the character defined in the Delimiter setting (default is a comma). If the user enters the delimiter character, it is interpreted as part of the text, and not as the delimiter.

Field as Section Header and Separator

You can configure fields as section headers and separators (white space) to group similar kinds of attributes and information.

There are two levels of section headers.

Section Header - Level 1

The screenshot shows a software interface for managing a customer widget. At the top, it displays the title "AEFQE_SBCorpCustomer Acme Widget rev 1: Properties". Below the title is a toolbar with various icons. A "Form Information" button is highlighted. The main area is divided into sections by vertical lines. The first section, "Section Header Level 1 - Customer Basics", contains fields for Type (AEFQE_SBCorpCustomer), Name (Acme Widget), Revision (1), Vault (AEF Host Vault3), Description (A Widget a Day Keeps the IRS Away), and Owner (AEF AllUser). The second section, "Section Header Level 2 - Customer Attributes", contains fields for SBAddrCity (Mumbai), SBAddrCountry (United States), and SBAddrState (CT).

Section Header Level 1 - Customer Basics	
Type	AEFQE_SBCorpCustomer
Name	Acme Widget
Revision	1
Vault	AEF Host Vault3
Description	A Widget a Day Keeps the IRS Away
Owner	AEF AllUser

Section Header Level 2 - Customer Attributes	
SBAddrCity	Mumbai
SBAddrCountry	United States
SBAddrState	CT

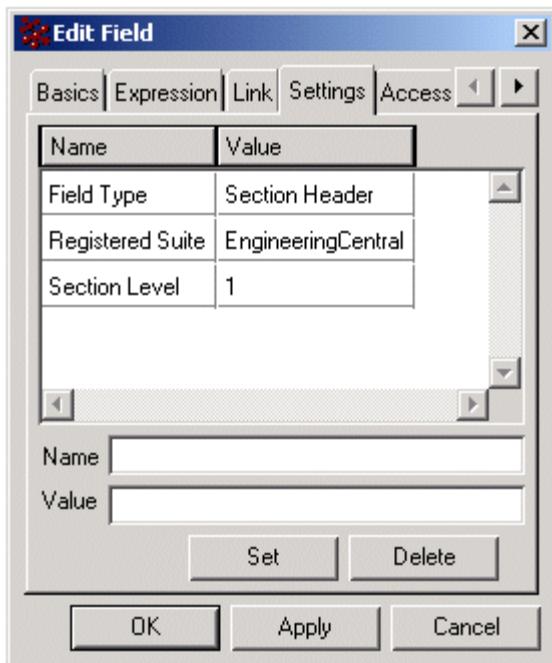
Section Header - Level 2

Section Separator

To configure a field to display a section header, use these parameters and settings:

- Field Type = Section Header
- Section Level = 1 or 2
- label=<Header Label>

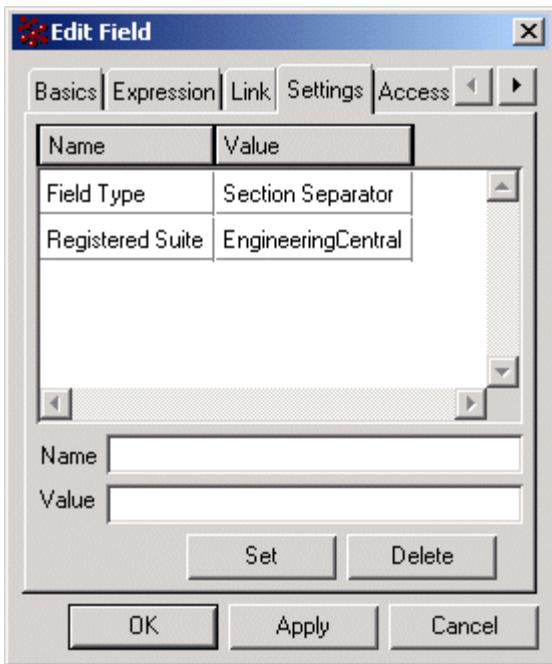
This dialog box shows the settings for configuring a field to display a section header.



To configure a field to display a section separator, use this setting:

- Field Type = Section Separator

This dialog box shows the settings for configuring a field to display a section separator.



Fields that are Grouped

The Group Name and Vertical Group Name settings allow you to control the layout of the web form. All fields with the same Group Name value show in the same horizontal row; all fields with the same Vertical Group Name value within a set of horizontally grouped fields show in the same column. The settings work in both View and Edit mode.

The following topics are discussed:

- [Horizontal Grouping](#)
- [Vertical Grouping](#)

Horizontal Grouping

You can configure a form to display a group of fields in one row using the Group Name setting. All fields with the same group name are grouped together.

Important: The "Group Holder" value for Field Type and the Group Count setting have been deprecated. To group fields on a form, you must use the Group Name setting.

For fields that should be grouped, add the setting Group Name=[name] to the field in the form component. All consecutive fields that have the same name will be considered one group. For example:

```
Field1  
Field2  
Field3 (Group Name = abc)  
Field4 (Group Name = abc)  
Field5 (Group Name = abc)  
Field6  
Field7 (Group Name = abc)  
Field8 (Group Name = abc)  
Field9 (Group Name = xyz)  
Field10 (Group Name = xyz)
```

For the above fields the form looks like the following

Field1		
Field2		
Field3	Field4	Field5
Field6		
Field7	Field8	
Field9	Field10	



Vertical Grouping

The Vertical Group Name setting works with the Group Name setting to allow some fields to span multiple rows within that group.

Policy	Thneed	Revision	1	State	Create Order
Type	Thneed				
Name	AEFQE_Thneed400000				
Vault	eService Production				
Owner	AEF AllUser				
Originated	Jan 30, 2008				
String	New	Description			
Thickness	0.00255 Cm	Real	0.01	Length	5.25 Meter
				Bool	TRUE

For fields that should be grouped, add the setting Vertical Group Name=[name] to the field in the form component. All consecutive fields that have the same name will be considered one group. This setting can be combined with the Group Name setting. For example:

```
Field1 (Group Name = abc) (Vertical Group Name = def)
Field2 (Group Name = abc) (Vertical Group Name = def)
Field3 (Group Name = abc) (Vertical Group Name = def)
Field4 (Group Name = abc) (Vertical Group Name = ghi)
Field5 (Group Name = abc) (Vertical Group Name = jkl)
Field6
```

For the above fields the form looks like this

Field1 Label	Field1 Value	Field4 Label	Field4 Value	Field5 Label	Field5Value
Field2 Label	Field2 Value				
Field3 Label	Field3 Value				
Field6 Label	Field6 Value				

Because Field4 and Field5 belong to different vertical groups, but are in the same horizontal group, they span all of the rows. You can also use the Hide Label=true setting with any field, and the multiple fields can be in any column; they do not need to be in the first column as shown in the example.

Fields Arranged in a Table

You can configure a form to display fields in a tabular format.

To arrange fields in a table, add a dummy field above the fields you want to include in the table. Define the dummy field using Field Type=Table Holder. It does not appear on the form. Using the Field Table Columns and Field Table Rows settings, you specify the number of columns and rows and their headings.

The system uses the first fields under the Table Holder field to populate the table. For example, if the settings specify 3 columns and 2 rows, the table contains 6 cells. Therefore, the system uses the first 6 fields under the Table Holder to populate the cells, working from upper left to lower right (the first field under the Table Holder is placed in the upper left cell, the second field is in the upper middle cell, and so on). After adding the Table Holder field, make sure the fields under it in the web form are the ones you want in the table and are in the correct order. The fields in a table do not need labels defined for them because the column and row headings serve as labels.

Attributes

Main Attributes	
Part Classification	Unassigned
Current State	Approved

Table Holder
defined with 3
columns and 2 rows

Cost and Weight Info

	Estimated	Target	Lead Time
Cost	0.0	0.0	Unassigned
Code	Make	Buy	Plastic

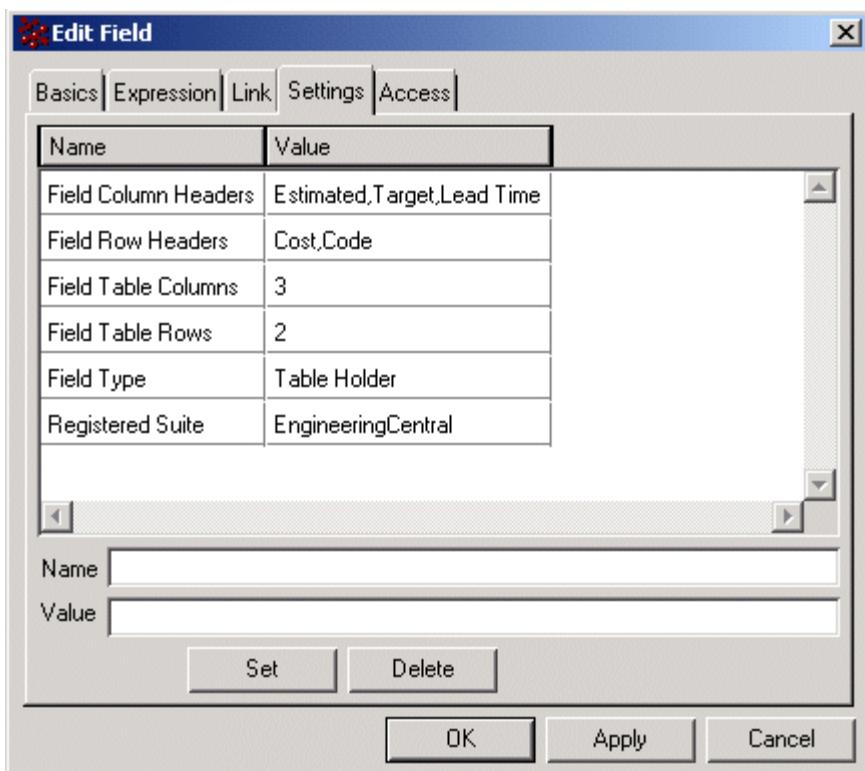
Column Header

Row Header

To configure a form to arrange fields in columns and rows, define a Table Holder field using these settings

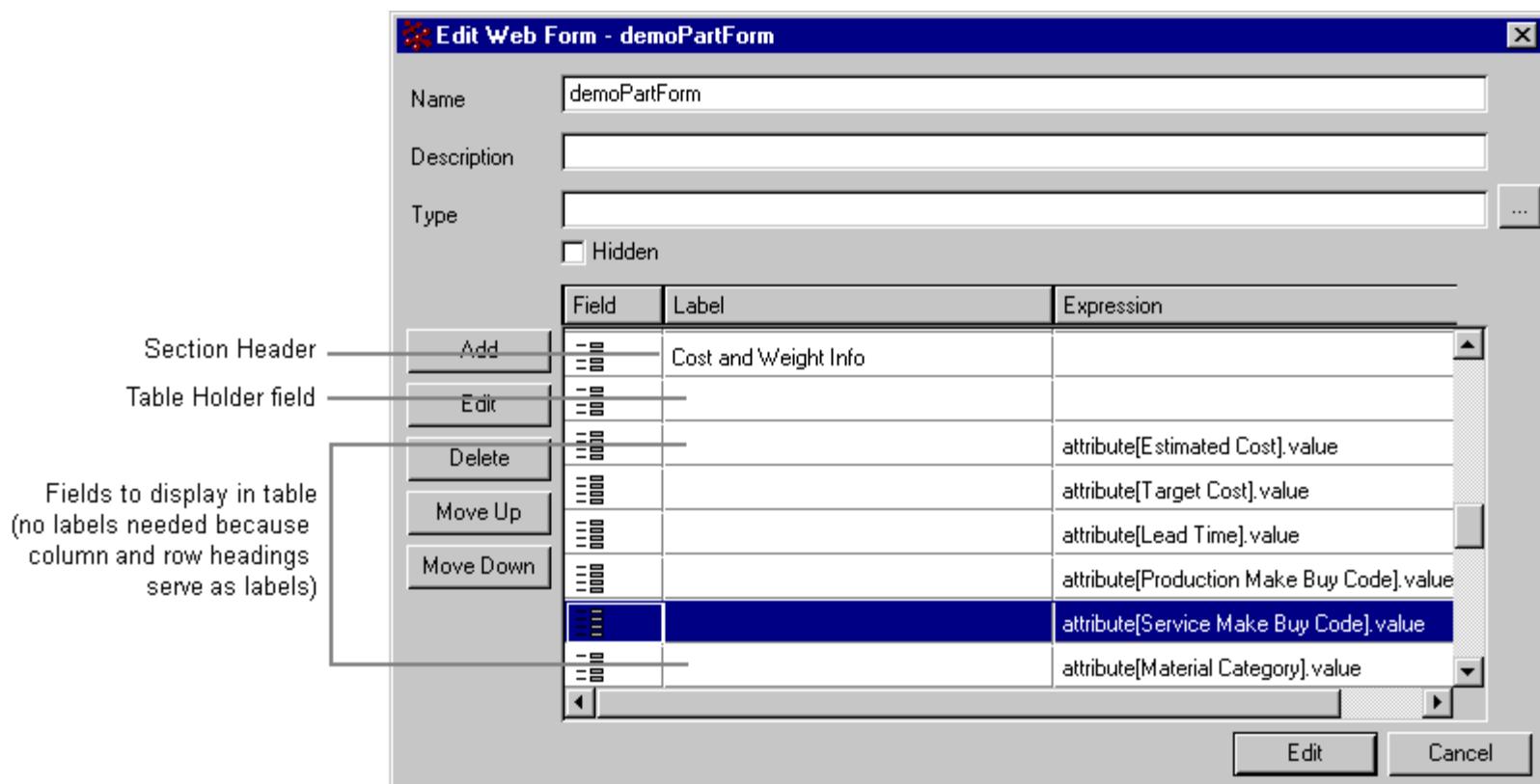
- Field Type = Table Holder
- Field Column Headers = Label for each column header, separated by a comma
- Field Row Headers = Label for each row header, separated by a comma
- Field Table Columns = 2, 3, ...
- Field Table Rows = 1, 2, 3, ...

This dialog box shows the settings to configure a field to serve as a table holder.



The row and column headers can also be string property keys. If a user does not have access to a field, then the table shows that cell as empty.

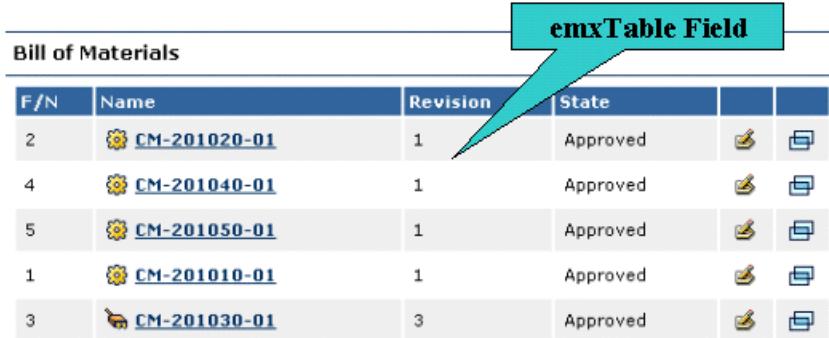
Make sure the fields under the Table Holder in the web form are the ones you want in the table. This dialog box shows how to configure a web form to display fields in a tabular format.



Field that Embeds a Configurable Table

You can configure a web form to display a configurable table.

Just add a dummy field to define the table using Field Type=emxTable.



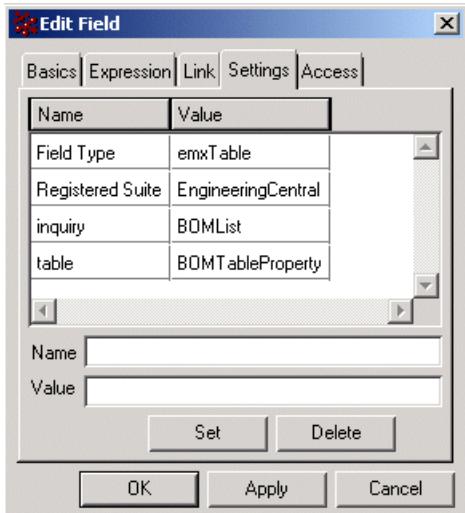
Bill of Materials					
F/N	Name	Revision	State		
2	 CM-201020-01	1	Approved		
4	 CM-201040-01	1	Approved		
5	 CM-201050-01	1	Approved		
1	 CM-201010-01	1	Approved		
3	 CM-201030-01	3	Approved		

To configure a form to display a configurable, use these settings

- Field Type = emxTable
- table = <name of the admin table>
- sortColumnName = <name of the column in table>
- Sort Direction = <ascending or descending>

Also include 1 of the following to get the list of objects:

- inquiry = <admin inquiry name>
- program = <JPO program name:JPO method name>



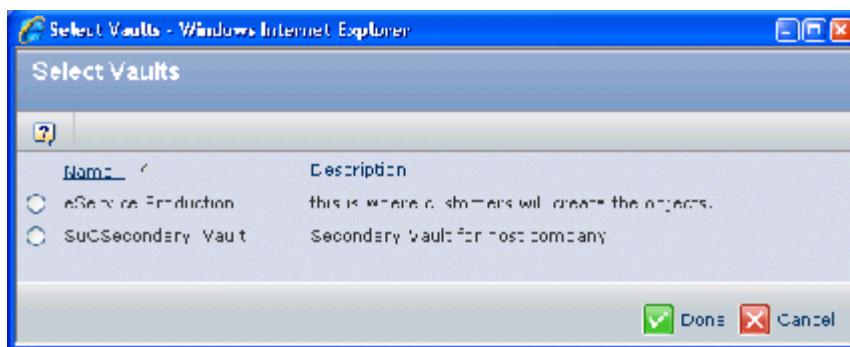
Field with Popup Range Helper

You can configure form fields or table columns so the Edit mode field displays a Browse button to help the user select the value to enter into the field.

Users select the range value from a user defined/custom popup window or from one of the common chooser components available in Business Process Services, such as the Type Chooser. This Vault field is an example of a field with a popup range helper.



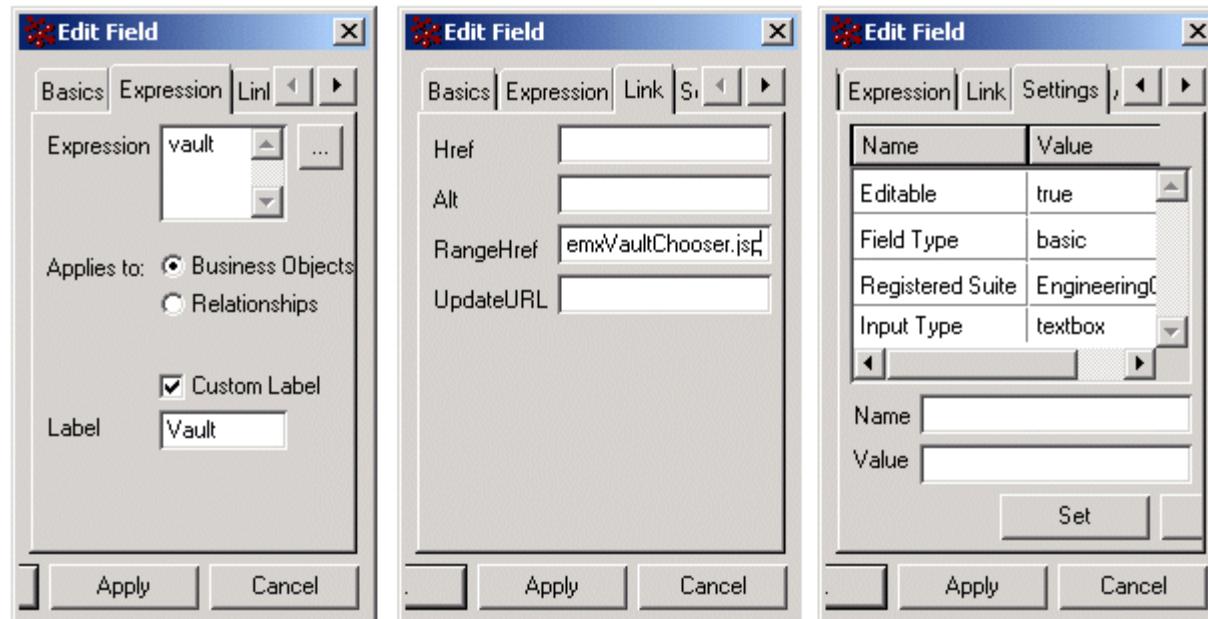
Users access the popup window using the Browse (...) button to the right of the text box. For example, the Vault field can be configured to display a Select Vault page as shown below.



To configure a field so that users can choose the value from a page in a popup window (only when the form is in edit mode), use these parameters and settings:

- RangeHref parameter--Enter the page that should appear in the popup window plus any parameters needed by the page. This page can either be a common chooser page included with the framework, such as the Vault chooser, or a custom selection page. The page must accept two parameters that the Edit mode form page appends to the URL automatically. For details on implementing the Range Helper page, see [Implementing Range Helpers for Choosers or Custom Pages](#).
- Input Type setting--Set to textbox. Since textbox is the default input type, you do not have to specify the setting if you want a text box.
- Editable setting--Set to true.

These dialog boxes show how to configure a field so users can select a range value from a popup window.



Fields with Dynamic Attributes

You can define a webform so that if an attribute is added to a type, either by customizing (using Business Modeler or MQL) or by installing a new release, the attribute will automatically appear on that form.

When one field on a webform has the setting Field Type=Dynamic Attributes, all attributes for that type are automatically added to the form.

You can exclude attributes from being dynamically added to the form using the View Exclude, Create Exclude, and Edit Exclude settings when defining the dynamic field.

When you use this field setting, the webform displays all fields explicitly defined, then the remaining attributes in alphabetic order within the dynamic attributes field(s). Fields that are explicitly added to a form are not added in the dynamic attributes fields. Hidden attributes are not added to the form. A dynamically-added field uses the attribute's default value if one is defined.

When using dynamic attributes, any attribute configured with a dimension will include the units of measure with the attribute value using this selectable:

```
attribute[ATTRIBUTE_NAME].inputvalue.tostring
```

Refer to [Selectables](#), for more information.

Attributes with units of measure will include the user's preferred system (Metric, English, or As Entered). If the defined units are the same as the preference settings, the user sees that value/unit. Otherwise, the preferred value/unit shows after the defined value/unit like this:

```
1.1 yd (1 m)
```

You can use the Category field setting to group attributes that have the uiform_Category property defined. For example, if an attribute has the property uiform_Category=EC Technical, use the setting Category=EC Technical as part of the field definition. The webform then pulls in all attributes with this value, and no other attributes unless explicitly defined.

You can define a dynamic field to show attributes with a Category, or a dynamic field to show all attributes for the object type. You cannot combine both on the same form.

The webform groups all attributes with the same category name into sections, and uses the Category value as a heading. You can add any number of categories to an attribute. If an attribute appears in more than one category on a form, editing it in one location automatically updates its value in every other location on the form.

To add properties to an attribute, use the MQL modify command to add the uiform_Category property and a value. For example, this MQL statement add the category EC Technical to the Weight attribute:

```
modify attribute Weight add property uiform_Category "EC Technical";
```

Refer to the [MQL Guide](#) for instructions on the modify command; refer to the [Business Modeler Guide](#) for instructions on defining form fields.

The Label setting lets you define a section heading for the web form other than the category value. You can also use a plain string as the Label value, but that string will not be internationalized. If you omit the Label setting or set the Label value as a blank, then no section heading displays on the webform.

To internationalize category values used as section headings, use the Label setting in conjunction with a string properties file. For example, if Category=NumericAttributes, you can include the setting Label=emxFramework.Title.NumericAttributes. The system looks up the value for this string in the appropriate language string file and uses it as the section heading. You also need to update the string properties file as described in the [Live Collaboration Administrator's Guide](#).

Sometime using dynamic attributes can result in adding several or dozens of fields to a form. To prevent the form from becoming too long, you can use the Column Count setting to arrange the fields across the page. The default Column Count is 1.

You can use Column Count on view and edit pages, but not create pages.

For example, a form definition includes these field definitions in addition to several explicitly defined fields:

```
Field Type=Dynamic Attributes,Category=Numeric,Column  
Count=2,Label=emxFramework.Title.NumericAttributes  
FieldType=Dynamic Attributes,Category=Date,Column  
Count=1,Label=emxFramework.Title.DateAttributes
```

These definitions add 2 sections to the webform, the first contains all attributes with the uiform_Category property of Numeric, and the other with the uiform_Category property of Date. For both section headings, the webform looks up the Label in the Framework string properties file.

AEFQE Sample Data|Create Thneed

Fields in red italics are required

Form Information			
Name	<input type="text"/>		
Type	Thneed		
Vault	eService Production		
Modified			
Numeric Attributes			
Int (Def)	<input type="text" value="3"/>	Real (Def)	<input type="text" value="47.008"/>
Date Attributes			
Date (Def)	<input type="text" value="Feb 20, 2006"/> <input type="button" value="..."/>		
<input checked="" type="checkbox"/> Done <input type="button" value="Cancel"/>			

This example shows an editable webform with dynamic attributes added to the Attributes for selected type (the uiform_Category property for the type, and the Category setting for the field):

AEFQE Sample Data|Create Layout Webform DA

Fields in red italics are required

Form Information					
Type	<input type="text" value="Part"/> <input type="button" value="..."/>				
Name	<input type="text"/>				
Policy	<input type="button" value="..."/>				
Owner	<input type="text" value="Test E"/>				
Attributes for selected type					
Current Version	<input type="text" value="0"/>	Design Purchase	Design <input type="button" value="..."/>	Effectivity Date	<input type="text"/> <input type="button" value="..."/>
End Item	No <input type="button" value="..."/>	End Item Override Enabled	Yes <input type="button" value="..."/>	Estimated Cost	<input type="text"/> <input type="button" value="..."/>
Is Version	FALSE <input type="button" value="..."/>	Lead Time	Unassigned <input type="button" value="..."/>	Material Category	Metal <input type="button" value="..."/>
Originator	<input type="text"/>	Part Classification	Unassigned <input type="button" value="..."/>	Production Make Buy Code	Unassigned <input type="button" value="..."/>
Service Make Buy Code	Unassigned <input type="button" value="..."/>	Sourcing Code	<input type="text" value="A-001"/>	Spare Part	No <input type="button" value="..."/>
Target Cost	<input type="text"/>	Unit of Measure	EA (each) <input type="button" value="..."/>	Weight	<input type="text"/> g <input type="button" value="..."/>
<input checked="" type="checkbox"/> Done <input type="button" value="Cancel"/>					

When displaying attribute fields on an create, view, or edit webform, the form uses default UI elements for the dynamically-added attributes:

Attribute Data Type	HTML Field Style	Validation
String	Text box	No validation

Integer/Real	Text box	Numeric validation
Multiline	Text area	No validation
Datetime	Read only text box with date picker	Date validation
Boolean	View mode: text string (TRUE or FALSE) Edit mode: checkbox	No validation (user must choose a valid value)
Range	Drop-down box (if enumerated values); Text box (non-enumerated values)	No validation

When setting a field type as Dynamic Attributes, these settings, if passed, are ignored:

Access Expression	Maximum Length
Access Function	OnChange Handler
Access Mask	OnFocus Handler
Access Program	Popup Modal
Admin Type	Printer Friendly
Allow Manual Edit	program
Alternate OID Expression	Range Function
Alternate Type expression	Range Program
Cols	Reload Function
Default	Reload Program
Display Format	Remove Range Blank
Display Time	Required
Edit Access Function	Rows
Edit Access Program	Section Level
Editable	Show Alternate Icon
Field Column Headers	Show Clear Button
Field Row Headers	Show Type Icon
Field Size	sortColumnName
Field Table Columns	Sort Direction
Field Table Rows	table
format	TargetLocation
function	Tip Page
Group Count	Update Function
Help Marker	Update Program
Hide Label	Validate
Image	Validate Type
Input Type	Window Height
inquiry	Window Width

Implementing Range Helpers for Choosers or Custom Pages

You can configure fields so that when the form or table is in Edit mode, users can click a Browse (...) button next to the field and then choose a range value from a page in a chooser or custom window.

The following topics are discussed:

- [Chooser as a Range Helper Page](#)
- [Custom User-Defined Page as a Range Helper](#)

Chooser as a Range Helper Page

This chooser page is specified in the RangeHref parameter for the field. See [Field with Popup Range Helper](#) for details on configuring this type of field. The page can be a common chooser page installed with the framework or can be a custom page. Implementation details for both types of range helpers are described below.

Note: You can also use a pre-defined type ahead chooser for Person, Organization, or Type fields. See [Automatic Type Ahead](#).

Business Process Services includes chooser pages, which can be used as range helper pages and specified as the RangeHref for a field. For example, the Type Chooser component can be used by entering the following in the RangeHref parameter:

```
 ${COMMON_DIR}/  
 emxTypeChooser.jsp?fieldNameActual=PartType&fieldNameDisplay=Pa  
 rtTypeDisplay&formName=PartList&SelectType=multiselect&Inclusio  
 nList=eServiceEngineeringCentral.Types&ObserveHidden=False&Suit  
 eKey=eServiceSuiteEngineeringCentral>ShowIcons=False
```

The chooser pages require two parameters shown below. The Edit mode of the form or table page automatically appends these two parameters to the RangeHref, so you do not need to add these parameters to the RangeHref parameter:

- fieldNameDisplay--the form or table element name to display the field display value (translated).
- fieldNameActual--the hidden form or table element to store actual field value (to be stored in database).

The chooser pages update the Edit form field when the user selects a new value.



Custom User-Defined Page as a Range Helper

To specify a custom select or chooser page as the range helper page, enter the page name plus any needed parameters in the RangeHref parameter for the field.

For example:

```
customRangeHelper.jsp?parameter1=value1
```

The custom JSP page must handle these parameters:

- fieldNameDisplay--the form or table element name to display the field display value (translated).
- fieldNameActual--the hidden form or table element to store actual field value (to be stored in database).

The Edit mode of the form or table page automatically appends these two parameters to the URL. So if the RangeHref parameter contains `customRangeHelper.jsp?parameter1=value1`, the resulting URL would be:

```
customRangeHelper.jsp?parameter1=value1&  
fieldNameDisplay=fieldDisplay&fieldNameActual=fieldName
```

When the user selects a new entry, the custom JSP must set these two elements. Use the following JavaScript to update the Edit form field:

```
var setString =  
 ("top.opener.document.frames[1].document.forms[0]['"+formFieldName+"'].value='"+newValue+"';");  
 eval(setString);
```

Validating Form Field Data

You can configure form fields so data is validated on the client side before the data is submitted to the server where the changes are committed.

For every form field, you can use either standard JavaScript validation or custom (user-defined) JavaScript validation. The `emxUIFormHandler.js` file contains JavaScript APIs that can be used for validation. This section describes both methods.

The following topics are discussed:

- [Standard JavaScript Validation](#)
- [User Defined / Custom JavaScript Validation](#)

Standard JavaScript Validation

Using the format field setting, you can have field data validated when it is displayed in Edit mode and the Editable setting is true. For example, if a field has the setting `format=integer`, when the user clicks Done on the Edit Form page, the system validates this field's value to make sure it is an integer value. The valid values for the format setting are:

- date
- real
- integer
- numeric

The field value is also validated when the field includes the `Required=true` setting. The system displays an appropriate JavaScript message in case of invalid entries.



User Defined / Custom JavaScript Validation

You can use a custom JavaScript method to validate any field when it is displayed in Edit mode and the Editable setting is true.

The `emxUIFormHandler.js` file contains these methods that can be used for validation:

- `emxFormGetValue`: Returns an object that holds the actual/display values for both the modified and old values of the field.
- `emxFormSetValue`: Sets the displayed field value and actual field value.
- `emxFormGetFieldHTMLDOM`: Gets the HTML DOM object for a field.
- `emxFormDisableField`: Enables or disables a field (gray or ungray).
- `emxFormIsFieldEditable`: Returns a boolean value indicating if the field can be edited.
- `emxFormSetFieldEditable`: Sends true to make the field editable; or false to make the field non-editable.
- `emxFormReloadField`: Reloads the specified field.

Refer to the JavaDocs for details on these functions.

Framework Validation FileBusiness Process Services

The Business Process Services validation file is defined using the following property:

```
eServiceSuiteFramework.UIForm.ValidationFile = scripts/  
emxUIFormValidation.js
```

Or for backward compatibility:

```
emxFramework.UIForm.ValidationFile = scripts/  
emxUIFormValidation.js
```

`emxUIFormValidation.js` is common and is available to all the web form pages. It is distributed with Business Process Services. Any Business Process Services upgrade overwrites it. Currently, the file is empty and serves as a placeholder for future validation methods that may be needed for all applications. Do not change this file since it will get overwritten in future upgrades.

To add any common custom files to be available across all web forms used by different applications, add the custom files to this property. For example:

```
eServiceSuiteFramework.UIForm.ValidationFile =scripts/  
emxUIFormValidation.js, \  
scripts/abcCustomValidation1.js
```

In this case `scripts/abcCustomValidation1.js` will not get overwritten by Business Process Services upgrades.

Application-Specific Validation File

You can define application-specific custom validation files using the property based on the Registered Suite (suite key) in `emxSystem.properties`. For example, if the Registered Suite is `EngineeringCentral`, the corresponding key is defined in the system property file as shown below:

```
EngineeringCentral.UIForm.ValidationFile = scripts/  
emxEngrValidation.js
```

Or:

```
eServiceSuiteEngineeringCentral.UIForm.ValidationFile =  
scripts/emxEngrValidation.js
```

You can assign one or more files separated by a comma as the property value to define multiple files.

Committing Changes Made on Edit Form

Clicking Done on the Edit mode form page commits all changes to the database. Clicking Cancel cancels all changes.

When the user clicks Done, the system executes the following:

1. Validates all the field values for the format specified in the field format setting. The system also checks for the presence of values in fields with the Required=true setting. For more information, see [Validating Form Field Data](#).
2. Validates all the field values using any user defined/custom JavaScript methods configured in the field Validate setting.
3. Submits the form for processing and checks to see if there are any changes made to the data.
4. Updates the changed fields by grouping them as business object basics, attributes, relationship attributes and program fields.

If the user clicks Done multiple times, multiple submissions are not sent.

You can also refresh the view mode after editing object properties.

After the system commits the form page (in Edit mode) changes to the database successfully, it refreshes the opener frame (typically the View mode of the form page) to display the updated data.

When the object's vault is changed, the objectId changes and therefore the objectId on the View mode of the form page updates automatically to refresh the data.

ENOVIA recommends that vault changes not be done. If business logic requires that the vault be changed, then it must be controlled in an action outside of the Edit Properties function to force a full tree refresh.

When the form is updated while the context tree is present, and the change caused an objectId change, then the tree is updated with the new objectId. The form component automatically senses the change in objectId and updates the context tree as required.

Generic Create Form

The generic create form component allows a single base JSP, emxCreate.jsp, to be used to create any object instead of having a separate JSP for each object.

In this section:

- [About the Generic Create Form](#)
- [Results of the Create Page](#)
- [Create Page Toolbar](#)
- [Defining the Basic Attributes](#)
- [Additional Fields on the Create Page](#)
- [Grouping Fields on the Create Form](#)
- [Parameters Supported by emxCreate.jsp](#)
- [URL Parameters Accepted by emxCreate.jsp](#)
- [Settings Supported by emxCreate.jsp](#)
- [Create Form Code Samples](#)

About the Generic Create Form

The generic create form component allows a single base JSP, emxCreate.jsp, to be used to create any object instead of having a separate JSP for each object.

This figure shows a sample create dialog page:

The screenshot shows a Windows Internet Explorer window titled "DocuSpace|Create Meeting". The window has a blue header bar with the title and standard window controls. Below the header is a toolbar with icons for back, forward, and help. A message box in the upper left corner says "Fields in red italics are required". The main area contains several input fields:

Name	<input type="text" value="autoName"/>
Type	<input type="text" value="Meeting"/>
Meeting Type	<input type="text" value="None"/>
Subject	<input type="text"/>
Meeting Location	<input type="text"/>
Context	Doc-Space
Description	<input type="text"/>
Meeting Date	<input type="text"/> <input type="button" value="..."/>
Start Time	<input type="text" value="8:30 AM"/> <input type="button" value="▼"/>

At the bottom right of the dialog are "Done" and "Cancel" buttons.

The URL to generate this page would be:

```
emxCreate.jsp?form=type_Meeting&type=type_Meeting&header=emxComponents.Common.CreateMeeting  
&Mode=create&helpMarker=emxhelpcreatemeting&preProcessJPO=emxMeeting:checkMeetingPrerequisites  
&submitAction=refreshCaller&nameField=both&postProcessURL=../components/emxMeetingProcess.jsp  
?action=CreateMeeting&typeChooser=true
```

To define a create page, use the guidelines for defining fields described in the following sections, and the tables that list the supported parameters, settings, and URL parameters that can be passed to emxCreate.jsp.

Many of the options supported by emxForm.jsp are also supported by emxCreate.jsp. For example, the type-ahead feature described in [Entering Field Values Using Type Ahead](#) works with the generic create component.

You cannot use the generic create component in portal mode.

Results of the Create Page

After the user completes the form and clicks Done, the system performs any specified validation defined by the Validate setting.

Validation for the generic create component works the same way as the web form component (emxForm.jsp). See [Validating Form Field Data](#) for more information.

Validation files can be specified in the `emxAPPNAME.properties` file, where `APPNAME` is the ENOVIA product name (for example, `emxEngineeringCentral.properties`). Use this property to specify the name of the validation file:

`emxAPPNAME.UIForm.Create.ValidationFile=filename`

where `filename` is the js or jsp file that executes the validation. The validation file must be stored in the corresponding application directory, such as `ematrix\engineeringcentral`. If a jsp file is defined, it cannot contain any `<script>` tags.

In addition, enter the method name as the value for the `Validate` setting on the webform field.

The system looks in the properties file for the application defined by the Registered Suite setting on the command that calls emxCreate.jsp to determine which validation file to use.

For example, if the `Registered Suite=Components` setting was set on the command that calls emxCreate.jsp, you could have this property set in `emxComponents.properties`:

`emxComponents.UIForm.Create.ValidationFile=createValidate.jsp`

and the `createValidate.jsp` must be stored in the `ematrix\components` directory.

If validation methods from more than one ENOVIA product need to be used on a single create form, all methods/files must be stored in the same directory (the one defined by the Registered Suite setting).

If validation fails, a message is displayed to the user. If validation passes, the create component makes any required connections.

After the object has been created, the value of the submitAction URL parameter determines if:

- A confirmation message displays (default behavior if the submitAction parameter is not passed):

`The following object was created successfully.`

`Type : <type>`

`Name : <name>`

`Revision : <revision>`

- The caller page gets reloaded.
- The new object's tree gets loaded in the main content frame.
- The new object's tree is loaded in a new popup window.

Create Page Toolbar

The toolbar on the generic create component contains specific items and is not configurable.

The toolbar contains these items:

- Help icon--opens the Create Component's help window
- Tip Page icon--opens the Create Component's tip page

The icons only display if the HelpMarker or TipPage URL parameters are passed to emxCreate.jsp.

You cannot add an Actions menu or other toolbar commands to the toolbar on the generic create component page.

Defining the Basic Attributes

To create any object within ENOVIA Live Collaboration, you must provide values for the basic attributes.

The following topics are discussed:

- [List of Basic Attributes](#)
- [Type Field](#)
- [Name Field](#)
- [Policy Field](#)
- [Vault Field](#)
- [Owner Field](#)

List of Basic Attributes

- Type
- Name
- Policy
- Vault
- Owner

These basic attribute fields display on the create page in the above order. If you configure any of these fields explicitly as described in [Form Fields](#), then those fields will display in the order in which you configured them. If you explicitly configure all of the basic attributes, then none of the generic fields display at the top of the create page.

If you explicitly configure any of these fields, you cannot use the Update Program and Update Function settings for those fields.

See [Parameters Supported by emxCreate.jsp](#) and [Settings Supported by emxCreate.jsp](#) for more details about the parameters and settings used to configure fields for the create page.



Type Field

To define the type of object being created, you can configure the page to create a specific type, or to allow the user to select the type of object to create.

To define a specific type of object:

- Pass a single value to the type parameter, such as type=type_Part
- Set typeChooser=false

If you pass multiple types but set the typeChooser to false, the first type passed in is used to create the object. When the typeChooser is false, Type is a read-only field.

If you set the typeChooser to true, then the browse button displays after the type field. When clicked, the typeChooser displays all passed types and their subtypes. The default type will be the first type passed into the create form.

You can restrict the user to a specific set of types. To do so:

- Pass a comma-separated list of types, using symbolic names
- set typeChooser=false

The type field is displayed as a combo box listing the passed-in types.

You can also explicitly configure a Type field in the form. Then, if you pass a type parameter, that type is the default value for the field; if you do not pass a type parameter, no value is entered in the field until the user enters it. When you configure a field, the type chooser is the one you configure.

You can configure the Type field as a combo box using a Range Program or Range Function setting. The Range Program/Function returns the list of types to populate the combo box. If you also pass a type parameter, that value is used as the default, otherwise the first type returned from the Range Program/Function is the default.

See the guidelines in [Form Fields](#) for more information on configured fields.

In all cases, the type chooser does not permit the user to select an abstract type, and does not show hidden types.



Name Field

The Name field supports these formats:

- autoName

- keyin (default)
- both

The autoName and Both options use an eService Number Generator object configured for the type of object being created. To select the eService Number Generator object, the system first searches on the specified type. If none are found, the system continues to search up the type hierarchy until one is found, and if none are found, uses the global name generator.

When nameField=keyin, the page shows a textbox for the name field and the user must type the name.

When nameField= autoName, then the create component checks whether or not an eService Number Generator object has been configured for the Type being created. The number of generator objects found determines how the Type field appears.

nameField=autoName

Number of eService Number Generator Objects Defined	Name Field Appearance
None	No name field shown; the global name generator is used to name the object
One	No name field shown; this eService Number Generator is used to name the object
More than one	Name field shows as a combo box listing the available naming generators

When using nameField=both, you can also provide a value for the autoNameChecked URL parameter. When autoNameChecked=true, the Autoname checkbox is already checked when the create form opens. When set to false (the default value), the checkbox is not selected when the create form opens.

When nameField=both, the page shows a type field for text entry and an autoName check box. If the user does not select the check box, the text entered in the text box is used to name the object. If the user selects the check box, then the create component uses this logic to name the object being created:

nameField=both and user selects autoName check box

Number of eService Number Generator Objects Defined	Name Generation
None	The global name generator is used to name the object and any text entered in the text box is ignored
One	The configured eService Number Generator is used to name the object and any text entered in the text box is ignored
More than one	Adds an autoName Series field below the Name field configured as a combo box listing available name generators and any text entered in the text box is ignored

To display the name field in a different location, configure at as described in "Configuring Automatic Business Object Naming" in the *Live Collaboration Administrator's Guide*, and use the values described above (autoName, keyin, or both).



Policy Field

If the type of object being created is governed by only one policy, then this field is not shown to the user and the governing policy is automatically used. If you want to show the field to the user anyway, configure the field as normal for a web form to show the policy in read-only mode. See the description of the Editable setting in [Settings for Fields in Web Form Objects](#) for instructions.

If the type of object can be governed by more than one policy, then the policy field shows as a drop-down list with the policy names. To choose a default policy, pass the policy parameter with the default policy name to the emxCreate.jsp page. If no policy is passed, then the drop-down defaults to the first policy in the list. The list only contains policies for which the context user has create access.



Vault Field

To configure the vault field on the create page, you have these options:

- No vault field
- Use the default vault chooser (emxVaultChooser.jsp)
- Explicitly configure a vault field

When the vaultChooser parameter is not passed, the field is not shown on the page and the create page uses the context user's default vault.

When vaultChooser=true, the page shows the ellipsis button that accesses the vault chooser. If a vault parameter is passed, that value is used as the default vault. If no vault parameter is passed, the context user's default vault is used as the default vault.

If you pass a vault name as a URL parameter and set vaultChooser=false, the specified vault shows as a read-only field.

You can also explicitly define the vault field to show the context user's default vault as a read-only value. Or, you can define the vault field as a drop-down list and use a range program to populate the list. See [Implementing Range Helpers for Choosers or Custom Pages](#) for more information.



Owner Field

Normally, the owner field does not display and the context user is used as the value for this field. If the owner of the object needs to be a user other than the context user, you can pass the owner parameter to the create page and that value shows as read-only.

If the user needs to select an owner, you must configure the Owner field to use the person chooser. In this case, the page uses the value passed by the owner parameter as the initial value for this field, or, if no value is passed, it uses the context user.

See [Implementing Range Helpers for Choosers or Custom Pages](#) for instructions on configuring the person chooser.

Additional Fields on the Create Page

You can add any supported field type to a create form.

In addition to the basic attributes described in [Defining the Basic Attributes](#), you can configure any other required fields as described in [Form Fields](#).

The following topics are discussed:

- [Connections](#)
- [Hidden Fields](#)
- [Default Values](#)

Connections

After creating an object, you may need to connect it to another object. For example, when an RFQ is created, you usually want to connect it to an RFQ Template object using the RFQ Template relationship.

To configure an object to be connected to another object:

1. Configure a field with the RangeHref parameter to show the chooser for selecting a template or parent or other required object.
2. Use the Update Program or Update Function setting to make the connection between the newly-created object and the template/parent object.

The create page first creates the object, then passes the necessary information to the JPO defined by the Update Program or Update Function setting where the connection is made. Both actions are done in a single transaction. See [Connecting to an Object Sample](#) for sample code to connect the newly-created object to an existing object.

As an alternative, you can pass the objectId, relationship, and direction URL parameters to emxCreate.jsp and the connection will be made as part of the creation transaction. This case requires that you know the objectId of the object to which the newly-created object will be connected.

The connection is only made if validation passes. The create component checks whether or not the relationship and objectID parameters were passed in. If so, the create component makes the required relationship connection.



Hidden Fields

You may need to store information in hidden fields during the creation process. In this case, you can use the `Field Type = programHTMLOutput` setting. The program should return the HTML code stored in the hidden fields needed by the create page.

When using a field type of `programHTMLOutput`, the JPO specified must return a string in XHTML format. See [programHTMLOutput Sample](#) for an example JPO.

As an alternative, you can pass required parameter/value pairs as URL parameters (see [URL Parameters Accepted by emxCreate.jsp](#)) and the JPO that uses those parameters/values (such as a postProcessJPO) can access them using the `requestMap`.



Default Values

When defining fields, emxCreate.jsp automatically uses the default value of the attribute as the default field value. If you want to specify a different default, set the `Default=value` setting. The default value setting overrides the attribute's default value.

Grouping Fields on the Create Form

Grouping allows multiple fields to display in a column, or for a single field to span multiple rows.

In addition to grouping fields horizontally as described in [Fields that are Grouped](#), the create form allows you to group fields vertically using the Vertical Group Name setting. This setting is supported for emxCreate.jsp; it is not supported for the edit/view forms.

The number of fields with the same Vertical Group Name determines the number of rows in the group.

For example:

- Field1 (Group Name = Gr1, Vertical Group Name = VG1)
- Field2 (Group Name = Gr1, Vertical Group Name = VG1)
- Field3 (Group Name = Gr1, Vertical Group Name = VG1)
- Field4 (Group Name = Gr1, Vertical Group Name = VG2)
- Field5 (Group Name = Gr1, Vertical Group Name = VG3)
- Field6

For the above fields, the form looks like the following

Field1 Label	Field 1	Field4 Label	Field 4	Field5 Label	Field 5
Field2 Label	Field 2				
Field3 Label	Field 3				
Field 6 Label	Field 6				

Field 6 is the next defined field for the form, and is not part of any horizontal or vertical group. You can also omit a label (Hide Label = true) for a field used in a horizontal or vertical group.

If the context user does not have access to any of the fields, the field is ignored and the horizontal and vertical grouping will still be maintained.

You can also define a tabular layout as described in [Vertical Grouping](#).

Parameters Supported by emxCreate.jsp

The following table defines the administrative parameters accepted by emxCreate.jsp.

Parameter	Description	Accepted Values/Examples
Expression	<p>Use to enter the select expression that gets the field data. This expression is applied to either the relationship or business object, as specified in the Applies To options (in Business Modeler). If the expression is a business object basic or an attribute that will be editable, make sure you include the Field Type setting with either the basic or attribute value.</p> <p>The Expression and Applies to options in the Business Modeler are equivalent to the businessobject and relationship MQL commands.</p> <p>To see an example of a field configured to use an expression, see Field Values as Select Expressions.</p>	<p>For business objects:</p> <p>type name current \$<attribute[attribute_Originator]. value></p> <p>For relationships:</p> <p>\$<attribute[attribute_FindNumber].value> \$<attribute[attribute_Qty.value></p>
Label	<p>Use to enter the text that displays as the field's label. Make sure Custom Label is checked (in Business Modeler) so the system gets the label you specify. If Custom Label is unchecked, the system uses the expression as the label.</p> <p>Either a string resource ID for the text string or the actual text string. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components.</p> <p>The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it does not find one, it displays the entered text.</p>	<p>emxEngineeringCentral.common.Name emxTeam.Common.ProjectName Description</p>
RangeHref (range MQL command)	<p>When configuring a form field in Business Modeler, the RangeHREF parameter is set on the Link tab.</p> <p>Use to configure a textbox that has a Browse (...) button that calls a chooser or custom window from which users can select a value to populate the textbox. This Owner textbox is an example with a range helper.</p> <p></p> <p>In the RangeHref parameter, specify the href URL to display the window. For example, the href might call a custom selection page.</p> <p>Range helpers are available only in Edit mode. Only textbox controls can be configured with a range helper. To specify the control type, set Input Type=textbox.</p> <p>To see an example of a field configured with a RangeHref, see Field with Popup Range Helper. Also see Implementing Range Helpers for Choosers or Custom Pages.</p>	<p>You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components.</p> <p>For example:</p> <pre> \${COMMON_DIR}/ emxSelectVault.jsp \${SUITE_DIR}/ emxSelectUser.jsp emxTypeChooser.jsp?typeList=type_Part,type_Document ../common/ emxTypeChooser.jsp?&selectType=multiselect &SelectAbstractTypes=true&InclusionList=eServiceEngineeringCentral.Types&observeHidden=true&ShowIcons=true </pre>

URL Parameters Accepted by emxCreate.jsp

You can use the URL parameters listed in this section to emxCreate.jsp. Parameters marked with an asterisk (*) are required.

URL Parameter	Description	Possible/Default Values
autoNameChecked	Used with nameField=both URL parameter, determines if the Autoname checkbox is checked when the form is displayed.	true false (default)
CreateMode	Used in a post-processing page to define application-specific logic. For example, Engineering Central may execute certain functions while Library Central executes others after creating an object. The value should identify the application, and should be whatever value is required by the post-processing program.	ENG LBC
CreateProgram	The name of the program and method that executes in addition to the object creation.	JPONAME:METHODNAME
direction	If a relationship is passed, the direction parameter specifies the direction of the relationship from the perspective of the object being created. If from, the relationship would be FROM the passed object TO the newly-created object. If to, the relationship would be FROM the newly-created object TO the passed object.	From (default) To
findMxLink	When true, show the mxLink icon/command button on the toolbar, which opens a search dialog box.	true (default) false
form	The web form administrative object name used to present this form page, such as emxCreate.jsp?form=SCSBuyerDesk	SCSBuyerDesk ECPartCreate
header	The header displayed in the header frame of the form. The value can be a string resource ID or the label itself.	emxFramework.Part.Create Create Part
HelpMarker	The help marker tag used to show the help window when the user clicks the context help icon on the toolbar.	emxhelppartcreate
nameField	Configures the type of name field on the form.	autoName keyin (default) both
objectId	If an OID is passed, the object created will be connected to this object using the relationship specified by the relationship parameter.	OID of a business object.
owner	Assigns a default owner to the object being created. If no value is passed, the context user is assigned as the owner.	Design Engineer Buyer
policy	The policy to assign to the type being created. You can use either the original or symbolic policy name, although symbolic names are recommended.	policy_Part policy_DifferentPart
postProcessJPO	Used to specify the post processing JPO	<JPO Name>:<Method Name>

	<p>program name and the method name to invoke after form processing and database update.</p> <p>This parameter is applicable only for edit mode.</p>	<p>For example: emxPart:processECO</p>
postProcessURL	<p>Specifies the name of the JSP executed during edit form post processing. The JSP is executed only after the form processing and database update complete.</p> <p>This parameter is applicable only for edit mode.</p>	<p><code> \${SUITE_DIR}/emxCustomPostProcess.jsp <AppDirectory>/emxCustomPostProcess.jsp</code></p> <p>For example: engineeringcentral/emxCustomPostProcess.jsp</p>
relationship	The name of the relationship to use to connect the object being created to the object passed by the <code>objectId</code> parameter. You can use either the original or symbolic relationship name, although symbolic names are recommended.	relationship_EBOM EBOM
showPageURLIcon	<p>When true,  shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false,  does not show in the toolbar.</p> <p>The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in <code>emxSystem.properties</code>.</p>	true false
submitAction	<p>Determines the action to take after creating the object (in addition to closing the form window):</p> <p>If no value is passed for this parameter, this alert displays to the user:</p> <p><code>The object Type <type name> Name:<object name> Rev:<revision> is created successfully.</code></p>	<p>refreshCaller: Reload the calling page. If called from a table or structure browser, then reloads the table or structure browser.</p> <p>treeContent: Load the newly-created object's tree in the main content frame.</p> <p>treePopup: Load the newly-created object's tree in a new pop-up window.</p>
TipPage	Specifies the Web page (html or jsp) to launch when the user clicks the Tip page toolbar button in the form header frame.	<p>Name of a custom URL page. <code>emxForm.jsp? form=ENCPart&TipPage=../myapplication/showMyTipPage.jsp?</code></p>
type*	A comma-separated list of types that can be created by this form. The type can be the original or symbolic name of the business object, although symbolic names are recommended.	type_Part type_RFQ Part type_Part,type_RFQ
typeChooser	Specifies whether or not the type chooser will be used for the Type field. If true, the default type chooser, <code>emxTypeChooser.jsp</code> , is used. Only the types defined by the type parameter will be selectable by the type chooser.	true false (default)
vault	The vault where the object being created will be stored. You can use either the original or symbolic value name, although symbolic names are recommended.	Gold vault_Gold
vaultChooser	Specifies whether or not the vault chooser will be used for the Vault field. If true, the default type chooser, <code>emxVaultChooser.jsp</code> , is used.	true false (default)

Settings Supported by emxCreate.jsp

The following settings are supported by emxCreate.jsp. These settings, except for Name Field, are also supported by emxForm.jsp. Following the table is a list of settings supported by emxForm.jsp but not supported for emxCreate.jsp.

The following topics are discussed:

- [Settings Table](#)
- [Settings Not Supported](#)

Settings Table

Setting	Description	Accepted Values/Examples
*Registered Suite	The application the field belongs to. The system looks for files related to the field in the registered directory for that application, which is specified in emxSystem.properties.	Set the value without any spaces, for example, EngineeringCentral or Framework. Set the value to the suite name as defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. If the suite name starts with "eServiceSuite" then you can skip this prefix and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite".
Access Expression Access Function Access Program	Used to control access to form fields For details, see User Access to UI Components . Do not use object-specific expressions. You can use non-object expressions, such as: <code>context.user.name == "Test Everything"</code>	<code>context.user.name == "Test Everything"</code>
Admin Type	Use to translate fields whose values are names of administrative objects or ranges of attributes. For example, suppose you are configuring a field that shows an object's current state and you want the state name to be translated. You would add this setting and set the value to State. The translations for administrative object names are stored in the emxFrameworkStringResource.properties files, as described in the <i>Live Collaboration Administrator's Guide</i> .	These keywords get the field values translated for the appropriate type name: <ul style="list-style-type: none">• Type• State• Role• Relationship• Policy• Group• Vault• Attribute (for translating the attribute name, not the range values) To translate attribute and range values, specify the symbolic name of the attribute, which starts with "attribute_" and is followed by the attribute name with no spaces. For example: <ul style="list-style-type: none">• attribute_UnitOfMeasure• attribute_PartClassification
Allow Manual Edit	When true, users can manually edit the form row for this field. Applicable only when the range parameter is set to a URL or when the setting format is assigned to date or for fields of type combobox. It is ignored in all other cases. When this setting is true, the Admin Type setting is ignored.	false (default)--Manual entry is not allowed. true--Manual entry is allowed.
Column Count	The number of name/value columns to draw horizontally. For Create forms, this setting applies only to a field defined with Field Type = Dynamic Attributes.	1 (default) 2 3 <i>n</i>
Default	If a field's value is empty or null and this setting is defined, the default	The default value you want to display. This can be a string resource key or the actual characters you want to fill in as the

	<p>value is displayed for the field.</p>	<p>default. The wildcard (*) can be used for search criteria fields.</p> <p>emxFramework.Common.default</p> <p>All</p> <p>*</p>
Dynamic URL	<p>When enabled, users can enter URLs or mxLink values and the values will display and function as hyperlinks.</p>	<p>enable (default)</p> <p>disable</p>
Editable	<p>Use to indicate whether the field is displayed as editable or read only.</p>	<p>true (default)--Users can edit the field.</p> <p>false--Users cannot edit the field.</p>
Field Column Headers	<p>Used in conjunction with the Field Type=Table Holder setting. Comma-separated list to specify the labels for the column headings. The number of labels should be the same as the value for the Field Table Columns setting.</p> <p>You can specify either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID.</p> <p>See Vertical Grouping.</p>	<p>Comma-separated list of column heading labels:</p> <p>Min,Max,Avg</p>
Field Row Headers	<p>Used in conjunction with the Field Type=Table Holder setting. Comma-separated list to specify the labels for the row headings. The number of labels should be the same as the value for the Field Table Rows setting and should be separated by a comma.</p> <p>You can specify either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID.</p> <p>See Vertical Grouping.</p>	<p>Comma-separated list of row heading labels:</p> <p>Weight,Volume</p>
Field Size	<p>Determines the width of a textbox input type field. The width is given in pixels except when Input Type is textbox or not set. In that case, its value refers to the (integer) number of characters.</p> <p>This setting is ignored if Field Type = programHTMLOutput.</p>	<p>Number of pixels, for example:</p> <p>30</p> <p>20 (default).</p>
Field Table Columns	<p>Used in conjunction with the Field Type=Table Holder setting. Defines the number of columns for the table.</p> <p>See Vertical Grouping.</p>	<p>2, 3, ...</p>
Field Table Rows	<p>Used in conjunction with the Field Type=Table Holder setting. Defines the number of rows in the table.</p> <p>See Vertical Grouping.</p>	<p>1, 2, ...</p>
Field Type	<p>This setting is used:</p> <p>To indicate that the field's data should be obtained from a program or image instead of an expression.</p> <p>When the field data is obtained from an expression, if the data is basic information or an attribute. The system needs to know whether a field's data is basic information or an</p>	<p>program--The values for this field are obtained from a program (JPO). The program and function name are required as settings.</p> <p>programHTMLOutput--Same as the program setting, except the field value output is in XHTML format. Field values are placed in the table cell between <td> and </td> tags. This setting ignores other field settings such as Show Type Icon, href, format, and Alternate OID expression.</p> <p>image--The field's value is the primary image associated with the business object.</p>

	<p>attribute in order to update the information correctly. Specifying whether the field is basic or an attribute is only required for fields that will be editable.</p> <p>To indicate the field is a dummy field that defines fields to display in a table or group.</p> <p>For more information on specifying field data, see Form Fields.</p> <p>These Field Type values that are supported by emxForm.jsp are NOT supported for emxCreate.jsp: emxTable, Table Holder, Group Holder, ClassificationPath, ClassificationAttributes.</p>	<p>basic--The field displays basic information for the business object. Basic information includes name, type, originated, policy, etc. Specifying basic as the field type is only needed when the field is editable. The only editable basic information is: type, name, revision, current, policy, description, owner, vault.</p> <p>attribute--The field displays values for an attribute on the business object, such as Originator or Weight.</p> <p>Section Header--Adds a new section heading between the form fields. The setting Section Level determines the heading level. See Field as Section Header and Separator.</p> <p>Section Separator--Adds white space to separate fields and sections.</p> <p>Dynamic Attributes--Displays all attribute/value pairs associated with the context object in the properties page.</p> <p>Table Holder--Arranges the fields under the field in columns and rows. Table Holder fields serve as dummy fields to define the fields to display in a table. Also see these settings: Field Column Headers, Field Row Headers, Field Table Columns, Field Table Rows.</p>
format	Use to configure read-only fields to show names using Fullname format. When Editable=true, this parameter is ignored.	user
function	<p>The name of the method to call within the JPO program specified in the program setting. This method within the JPO is used to get the field values if the setting "Field Type" is set to "program" or "programHTMLOutput".</p> <p>For more information, see Field Values Obtained from a Program.</p>	<p>The name of a function in the program JPO, such as:</p> <p>getAssignedBuyerDesk getPackageAccess getParentPart</p>
Group Name	Used for grouping fields horizontally in web forms. The consecutive fields with same group name are considered a group.	The name of the horizontal group.
Hide Label	Displays or hides the label for a particular row on a form.	True False
Image	<p>Use to specify an image file when the field value should be an image only. This setting is required when the Field Type setting is set to image. This file must exist in the application server (not in the database). You can make the image a hyperlink by including a URL in the href parameter.</p> <p>To see an example of a field with an image, see Field Values as Hyperlinked Image.</p>	images/newPart.gif images/EditItem.gif
Input Type	<p>Specifies the type of HTML control to display for user input.</p> <p>Although multiple choices can be displayed in the create page, only one selection can be saved. To disable the ability to make multiple selections during view of webform, use programHTMLOutput and the html tag that does that.</p> <p>Use the radiobutton or combobox input types for fields that require a single selection.</p> <p>If you want to save multiple</p>	<p>textbox-Default. Provides a single-line box for typing text. This graphic shows a field configured as a Text Box with the Required setting equal to true.</p> <p>Weight <input type="text" value="0.0"/></p> <p>textarea-Provides a multi-line box for typing text.</p> <p>Description </p> <p>checkbox-Provides a check box next to each range value.</p>

	<p>selections, a custom JPO needs to be written using the checkbox or listbox input types which might delimit the choices in the attribute using the Studio Customization Toolkit.</p>	<p>Make Task Owner? <input type="checkbox"/></p> <p>See Sample JPO for Web Form with Custom Combobox.</p> <p>listbox-Provides a list of range values. Although users can select more than one item in the list by holding the Ctrl key, only one value will be saved unless a custom JPO is implemented.</p> <p>radiobutton-Shows a radio button next to each range value. To see an example of a field configured with radio buttons, see Field for Attribute with Choices in Radio Buttons.</p>
		<p>Unit of Measure</p> <ul style="list-style-type: none"> <input type="radio"/> LB (pound) <input type="radio"/> IN (inch) <input type="radio"/> GA (gallon) <input type="radio"/> FT (feet) <input checked="" type="radio"/> EA (each)
		<p>combobox-Provides a drop-down list of options and users can only select one value. Use combo boxes for attributes that have defined ranges and for attributes whose ranges are determined with a Range Helper URL. To see an example of a field with a combo box, see Field for Attribute with Choices in Combo Box.</p>
		<p>ECR Evaluator <input type="button" value="Test ECREvaluator"/> <input type="button" value="▼"/></p>
Maximum Length	Limits the number of characters that can be entered in a field with Input Type of textbox. If not specified, it uses the HTML default (unlimited).	Number of characters, for example: 30
Name Field	Defines how the name field for the object will display in the form:	autoName-only the autoName option is available keyin-a textbox is provided for text entry both-both a textbox and the autoName checkbox shown for the name field
program	<p>Use to specify the name of the JPO program to get the field's data. This program gets the field values when the <code>Field Type</code> setting is <code>program</code> or <code>programHTMLOutput</code>.</p> <p>Using a JPO to populate field data is recommended only when a select expression cannot be used to obtain the field value.</p> <p>To see an example of a field that uses a program, see Field Values Obtained from a Program.</p>	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
Range Function	Use to specify the name of the method in the JPO specified in the Range Program setting. See the Range Program setting below for more information.	The name of a function in the Range Program JPO, such as: getAssignedRange getClassificationRange getPartUOM
Range Program	<p>Use to specify the name of a JPO that contains a method to get the field value ranges (choices). A range program is used only when the Input Type setting is combobox, radiobutton, or checkbox.</p> <p>If the choices need to be presented on a page in a popup window, for example in a chooser, use the RangeHref parameter instead.</p> <p>To see an example of a field configured with a range program, see Field for Attribute with Choices in</p>	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm

	Combo Box . Also see JPO for Getting Field Range Values (Choices).	
Required	Use to indicate the value for this field is required.	true--When completing the form, the user must enter a value for the field. The field label appears in red italic text. If the user does not enter a value and clicks Done, a JavaScript message appears that prompts the user to enter a value. false (default)--When completing the create form, the user can leave this field blank.
Section Level	Used in conjunction with the Field Type: Section Header setting to define the level of heading. To see an example, see Field as Section Header and Separator .	Two heading levels are available: 1 (default)--Font for heading label is large and a horizontal line is included above the label. 2--Font for heading label is smaller and the heading is in the same gray rectangle as standard fields.
Show Clear Button	Adds a Clear hyperlink next to the field. This setting applies to textarea, textbox, chooser, and date picker input fields. When the user clicks the Clear link, it clears the content of the field.	true false
Show Type Icon	This setting applies only for fields that display the business type name in read-only mode. If set to true, the field value displays along with the type icon for the business object being displayed by the current form page. The icon is defined in the emxFramework.smallIcon property in emxSystem.properties. The icon displays to the left of the field data. If no icon is defined for this type, the system looks for a property defined for the parent type and so on up the hierarchy. If no property is defined for any type in the hierarchy, the system uses the default icon specified in the emxFramework.smallIcon.defaultType property.	true false (default) To see an example of a field with a type icon, see Field Values as a Hyperlink and Type Icon .
Update Function	Specifies a method name in the JPO given in the Update Program setting. See Update Program for more information. This setting can not be used with the required fields: Type, Name, Policy, Vault, Owner.	The name of a function in the Update Program JPO, such as: setAssignedBuyerDesk setPackage Access setPartClassification
Update Program	Specifies a JPO that contains a method to set the field value when the user clicks Done. This program is used only when the Field Type setting is program or programHTMLOutput. This setting can not be used with the required fields: Type, Name, Policy, Vault, Owner. Using an update program is recommended only when the Field Type is not attribute or basic. See JPO for Getting Field Values .	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
Validate	Specifies the method to execute for validating any cell. The file containing this method must be specified in the emxAPPNAME.properties file. See	The name of a method, such as: checkUniqueName

<u>Results of the Create Page.</u>		
Validate Type	Specifies how field values should be validated.	Basic or Restricted--The field values are validated against the value of emxFramework.Javascript.BadChars. Name--The field values are validated against the value of emxFramework.Javascript.NameBadChars.
Vertical Group Name	Used for grouping fields vertically in create web forms. The fields with the same Vertical Group Name value will be considered a group.	The name of the vertical group.

*Required Setting



Settings Not Supported

For reference, these settings are supported by emxForm.jsp, but are NOT supported by emxCreate.jsp:

- Access Mask
- Alternate OID Expression
- Alternate Type Expression
- Cols
- Display Format
- Display Time
- Group Count
- Help marker
- inquiryPopup Modal
- Printer Friendly
- Remove Range Blank
- Rows
- Show Alternate Icon
- sortColumnName
- sortDirection
- sort range values
- table
- Target Location
- Tip Page
- Window Height
- Window Width

Create Form Code Samples

This section provides code samples for working with the create form.

The following topics are discussed:

- [programHTMLOutput Sample](#)
- [Connecting to an Object Sample](#)

programHTMLOutput Sample

This sample shows Java code that can be used with a field type of programHTMLOutput. The code returns a string in XHTML format.

```
public String getObject(matrix.db.Context context, String []
args)
throws Exception
{
    HashMap paramMap = (HashMap)JPO.unpackArgs(args);
    HashMap requestMap = (HashMap)paramMap.get("requestMap");
    String objectId=(String)requestMap.get("objectId");
    MQLCommand objMQL = new MQLCommand();
    objMQL.open(context);
    String sMQLStatement = "print bus "+ objectId +" select name
dump |;";
    String objName = MqlUtil.mqlCommand(context,objMQL,
sMQLStatement);
    objMQL.close(context);
    StringBuffer output=new StringBuffer();
    output.append("<a
href=\"emxForm.jsp?form=type_Part&objectId=");
    output.append(objectId);
    output.append("\">
");
    output.append(objName);
    output.append("</a>");
    return output.toString();
}
public String drawRadioButton(matrix.db.Context context, String
[] args)
{
    StringBuffer output=new StringBuffer();
    output.append("<input type=\"radio\" name=\"gender\"
value=\"male\"> Male </input>");
    output.append("<br><br>");
    output.append("<input type=\"radio\" name=\"gender\"
value=\"female\"> Female </input>");
    return output.toString();
}
```



Connecting to an Object Sample

This sample shows an example of Java code to connect the newly-created object to another object.

```
public Object createAndConnect(Context context, String[] args)
throws Exception
{
    HashMap programMap = (HashMap) JPO.unpackArgs(args);
    HashMap paramMap = (HashMap) programMap.get("paramMap");
    String objectId = (String) paramMap.get("objectId"); //Part ID
    String newRDOIId = (String) paramMap.get("New OID"); // RDO Id
    String strNewPartPartRevisionRelationship = "Design
Responsibility";
    //Connect the part to RDO
    if (!newRDOIId.equals(" "))
    {
        setId(newRDOIId);
        DomainObject domainObjectToType = newInstance(context,
```

```
objectId); //Part Object
    DomainObject domainObjectFromType =
newInstance(context, newRDOIid); //RDO Object
DomainRelationship.connect(context, domainObjectFromType, strNewP
artPartRevisionRelationship, domainObjectToType);
}
return new Boolean(true);
}
```

JPO Interface for Form Fields

When you cannot accomplish a task or business requirement using a select expression or other feature supported by the configurable form component, you can use a JPO.

In this section:

- [About Configurable Form Support for JPOs](#)
- [JPO Method Signature and Input Argument](#)
- [JPO for Getting Field Range Values \(Choices\)](#)
- [JPO for Getting Field Values](#)
- [Update Methods In JPO Programs](#)
- [Sample JPO for Getting Field Values](#)
- [Sample JPO for Web Form with Custom Combobox](#)

About Configurable Form Support for JPOs

When you cannot accomplish a task or business requirement using a select expression or other feature supported by the configurable form component, you can use a JPO.

The configurable form supports JPOs to:

- Get the field values to be displayed in a form field
- Get the range values (choices) to be populated for a form row
- Update field values after updating the form

To use a JPO, you must specify parameters and settings in the field of the web form administrative object as described in [Parameters for Web Form Objects](#). The JPOs must follow the specific signatures described in the following section.

To see a sample JPO, go to [Sample JPO for Getting Field Values](#).

JPO Method Signature and Input Argument

The JPO class must have the mxMain method implemented. The JPO methods are defined with the following signature.

```
public static Object methodName(matrix.db.Context context,
String[] args)
throws Exception
{
}
```

The input arguments are:

- Context--passed in always.
- String[] args--will have one element of type HashMap.

The input argument "arg" is a HashMap, which contains the details of the object to be processed. The data structure of this input parameter (HashMap) is defined below:

Key Name	Data Type	This key is assigned to the:
languageStr	String	Language string for the current browser setting. The program can use this key for any internationalization purpose.
New Value	String	Applicable only for use with Update Program and Update Function. Key "New Value" is assigned to the new value changed by the user.
objectId	String	Business object Id (OID) to be used in the current form page.
Old Value	String	Applicable only for use with Update Program and Update Function. Key "Old Value" is assigned to the actual value for the field before the user changed it.
relId	String	Relationship id (RelID) to be used for the form page.
requestMap	HashMap	Key "requestMap" is assigned to a HashMap, which contains a set of key/value pairs as available in the request object. The keys are of type String (parameter names) and the values are of type String (parameter value).

The JPO method can extract the information from the input argument for processing the data.

```
public static Object methodName(matrix.db.Context context,
String[] args) throws Exception
{
HashMap programMap = (HashMap) JPO.unpackArgs(args);
HashMap requestMap = (HashMap) programMap.get("requestMap");
HashMap paramMap = (HashMap) programMap.get("paramMap");
String objectId = (String) requestMap.get("objectId");
String relId = (String) paramMap.get("relId");
String languageStr = (String) requestMap.get("languageStr");
?
return object
}
```

The method returns an object of appropriate type depending on the where it is used. The details of return types are explained in the following sections.

JPO for Getting Field Range Values (Choices)

When you use a JPO to get field value ranges, you must configure the form field with specific settings.

- Input Type=combobox / radiobutton / checkbox
- Range Program=JPO name
- Range Function=JPO method name

For processing the data, the input parameter can be used by the method. The return type of this method is a HashMap, which contains two StringLists: one for the range values and the other for internationalized range values in the same sequence.

The HashMap should look like the following:

```
Key Value
-----
field_choices range -> Values StringList
field_display_choices -> Internationalized range Values StringList
The JPO function template is provided below:
public static Object methodName(Context context, String[] args) throws Exception
{
    HashMap programMap = (HashMap) JPO.unpackArgs(args);
    HashMap requestMap = (HashMap) programMap.get("requestMap");
    HashMap paramMap = (HashMap) programMap.get("paramMap");

    // Get the required parameter values from "programMap" - as required
    String objectId = (String) paramMap.get("objectId ");
    String relId = (String) paramMap.get("relId ");
    String languageStr = (String) paramMap.get("languageStr");
    // initialize the return variable HashMap tempMap = new HashMap();
    HashMap tempMap = new HashMap();
    // initialize the Stringlists fieldRangeValues, fieldDisplayRangeValues
    StringList fieldRangeValues = new StringList();
    StringList fieldDisplayRangeValues = new StringList();
    // Process information to obtain the range values and add them to fieldRangeValues
    // Get the internationalized value of the range values and add them to
    fieldDisplayRangeValues
    fieldRangeValues.addElement(xxx);
    fieldDisplayRangeValues.addElement(internationalised xxx);
    tempMap.put("field_choices", fieldRangeValues);
    tempMap.put("field_display_choices", fieldDisplayRangeValues);
    return tempMap;
}
```

JPO for Getting Field Values

When you use a JPO to get field values, you must configure the form field with the settings described in this section.

- Field Type=program
- program=JPO name
- function=JPO method name

For processing the data, the input parameter can be used by the method. The return type of this method is a `StringList`, which contains one or more field value to be displayed.

Here is a template for this kind of JPO method:

```
public static Object methodName(Context context, String[] args)
throws Exception
{
    HashMap programMap = (HashMap) JPO.unpackArgs(args);
    HashMap requestMap = (HashMap) programMap.get("requestMap");
    HashMap paramMap = (HashMap) programMap.get("paramMap");
    String objectId = (String) requestMap.get("objectId");
    String relId = (String) paramMap.get("relId");
    String languageStr = (String) requestMap.get("languageStr");
    // initialize the return variable
    StringList fieldValues = new StringList();
    // define and add selects if required
    // Process the information to obtain the values
    if ( objectId != null )
    {
        //add the field values
        fieldValues.addElement(?);
    }
    return fieldValues;
}
```

Update Methods In JPO Programs

There are some update methods in JPO programs for configurable form fields that are using the requestMap to get the changed value of the field using the field name.

```
String[] revisionValues=(String[]) requestMap.get("Revision");
String attributeValue=(String) requestMap.get(attrName);
```

If you intend to use the method to work with an editable table component or structure browser, then you must code it differently using the paramMap:

```
String newValue = (String) paramMap.get("New Value");
```

If the programmer expects multiple values for the field then use the following code:

```
String[] newValues = (String[]) paramMap.get("New Values");
```

One exception where the requestMap can be used is when the field or column type is of programHTMLOutput. In this instance, it can be useful to get the values of the custom html fields given by the program.

The JPOs used for updating the Web form component fields should use requestMap to get the timeZone, charSet, localeObj and languageStr values. For example:

```
String timeZone = (String) requestMap.get("timeZone");
String charSet = (String) requestMap.get("charSet");
Locale localeObj = (Locale) requestMap.get("localeObj");
String languageStr = (String) requestMap.get("languageStr");
```

Sample JPO for Getting Field Values

The following is a sample Java Program Object (JPO) for getting the current vault for the object the form page currently pertains to.

You can use this type of JPO when the Field Type setting for a form field is set to program or programHTMLOutput. The getVault method processes the object and returns the vault value.

You can obtain the vault name by configuring the form field as a businessobject select expression "vault" instead of defining the Field Type=program. The "vault" example is used for simplicity and to illustrate the steps involved in writing the JPO for a form field.

```
/*
 * emxUIFormSample
 *
 * Copyright (c) 1992-2003 MatrixOne, Inc.
 *
 * All Rights Reserved.
 * This program contains proprietary and trade secret information of
 * MatrixOne, Inc. Copyright notice is precautionary only and does
 * not evidence any actual or intended publication of such program.
 *
 * static const char RCSID[] = $Id: Exp $
 */
import matrix.db.*;
import matrix.util.*;
import java.io.*;
import java.util.*;
import com.matrixone.framework.beans.*;
import com.matrixone.framework.util.*;
import com.matrixone.framework.ui.*;
/**
 * @version AEF 9.5.0.0 - Copyright (c) 2002, MatrixOne, Inc.
 */
public class ${CLASSNAME}
{
    /**
     *
     * @param context the eMatrix <code>Context</code> object
     * @param args holds no arguments
     * @throws Exception if the operation fails
     * @since AEF 9.5.0.0
     * @grade 0
     */
    public ${CLASSNAME} (Context context, String[] args)
        throws Exception
    {
        if (!context.isConnected())
            throw new Exception("not supported on desktop client");
    }
    /**
     * This method is executed if a specific method is not specified.
     *
     * @param context the eMatrix <code>Context</code> object
     * @param args holds no arguments
     * @returns nothing
     * @throws Exception if the operation fails
     * @since AEF 9.5.0.0
     */
    public int mxMain(Context context, String[] args)
        throws Exception
    {
        if (!context.isConnected())
            throw new Exception("not supported on desktop client");
        return 0;
    }
}
```

```

}

/**
 * get Vault for the object.
 *
 * @param context the eMatrix <code>Context</code> object
 * @param args holds the following input arguments:
 *          0 - HashMap programMap
 * @returns StringList containing Vault name
 * @throws Exception if the operation fails
 * @since AEF 9.5.0.0
 */
public static Object getVault(Context context, String[] args)
    throws Exception
{
    HashMap programMap = (HashMap) JPO.unpackArgs(args);
    String objectId = (String)programMap.get("objectId");
    StringList vaultList = new StringList();
    StringList listSelect = new StringList(1);
    listSelect.addElement("vault");
    if ( objectId != null )
    {
        BusinessObject bo = new BusinessObject(objectId);
        bo.open(context);
        String vault = bo.getVault();
        bo.close(context);
        vaultList.addElement(vault);
    }
    return vaultList;
}
/**
 * set Vault for the objects.
 *
 * @param context the eMatrix <code>Context</code> object
 * @param args holds the following input arguments:
 *          0 - HashMap programMap
 * @returns vector of Vault names
 * @throws Exception if the operation fails
 * @since AEF 9.5.0.0
 */
public static int setVault(Context context, String[] args)
throws Exception
{
    // Map containing the request parameters
    HashMap requestMap = (HashMap) programMap.get("requestMap");
    // Map containing the key parameters
    HashMap paramMap = (HashMap) programMap.get("paramMap");
    String objectId = (String)paramMap.get("objectId");
    String newValue = (String)paramMap.get("New Value");
    if ( objectId != null && newValue != null )
    {
        // Vault newVault = new Vault(newValue);
        BusinessObject busObj = new BusinessObject(objectId);
        busObj.open(context);
        // busObj.setVault(context, newVault);
        BusinessObject newBusObj = busObj.change(context, busObj.getTypeName(),
busObj.getName(), busObj.getRevision(), newValue, busObj.getPolicy().toString());
        busObj.close(context);
    }
    return (0);
}

/**
 * get getOriginatorRange for the field.
 *
 * @param context the eMatrix <code>Context</code> object

```

```
* @param args holds the following input arguments:  
*      0 - HashMap programMap  
* @returns StringList of Range values  
* @throws Exception if the operation fails  
* @since AEF 9.5.0.0  
*/  
  
public static Object getOriginatorRange(Context context, String[] args)  
    throws Exception  
{  
    HashMap programMap = (HashMap) JPO.unpackArgs(args);  
    String objectId = (String)programMap.get("objectId");  
    // HashMap paramMap = (HashMap)programMap.get("paramList");  
    System.out.println("objectId << " + objectId + " >>");  
    StringList rangeList = new StringList();  
    rangeList.addElement("Test SeniorDesignEngineer");  
    rangeList.addElement("Test Everything");  
    rangeList.addElement("Test DesignEngineer");  
    rangeList.addElement("Test Buyer");  
    System.out.println("rangeList << " + rangeList + " >>");  
    return rangeList;  
}  
}
```

Sample JPO for Web Form with Custom Combobox

This section provides a sample JPO for a web form that uses a custom combobox.

```
import matrix.db.*;
import matrix.util.*;
import java.io.*;
import java.util.*;
public class ${CLASSNAME}
{
    public ${CLASSNAME} (Context context, String[] args) throws Exception
    {
    }

    public String getChoicesHTML(Context context, String[] args) throws Exception
    {
        // check String[] args passed in to tell if view or edit?
        System.out.println("Array length = " + args.length);
        HashMap myMap = (HashMap) JPO.unpackArgs(args);
        HashMap requestMap = (HashMap) myMap.get("requestMap");
        /*
        System.out.println("\ntags within HashMap.....: " + requestMap + "\n");
        */
        //Edit mode:
        // this should return "edit" if "edit" then build combobox else just display value for
        view mode
        String edit = (String) requestMap.get("mode");
        String sReturn = "";
        if (edit == null ) // display value for view mode in webform
        {
            sReturn = "3";
            System.out.println("mode not edit");
        }
        else // display list of values in combo box for edit mode
        {
            System.out.println("mode IS edit");
            sReturn = "<SELECT name=\"test\"> <OPTION>1 <OPTION>2 <OPTION SELECTED>3 <OPTION>3
<OPTION>4 <OPTION>5 </SELECT>";
        }
        // Your webform field needs these settings
        // Field Type=programHTMLOutput
        // function=JPOMethodName
        // program=JPOName
        // Watch the spelling and case of the above settings!
        // To actually save the value selected you would have to add the following settings with
        a new save method
        // Update Function=JPOMethodName
        // Update Program=JPOName
        // the data to be save will be in the requestMap similar to above
        // this returns the String needed to display the custom HTML combo with 3 SELECTED as
        default
        return sReturn;
    }

    public static Object getChoices(Context context, String[] args) throws Exception
    {
        System.out.println("inside getChoices2 method");
        HashMap programMap = (HashMap) JPO.unpackArgs(args);
        StringList fieldRangeValues = new StringList();
        // Get the required parameter values from "programMap" - as required
        String objectId = (String) programMap.get("objectId ");
        String relId = (String) programMap.get("relId ");
        String requestQueryString = (String) programMap.get("requestQueryString");
        String languageStr = (String) programMap.get("languageStr");
        fieldRangeValues.addElement("test2_1");
    }
}
```

```
fieldRangeValues.addElement("test2_2");
fieldRangeValues.addElement("test2_3");
fieldRangeValues.addElement("test2_4");
fieldRangeValues.addElement("test2_5");
System.out.println("done adding elements");
return fieldRangeValues;
}
}
```

Tables

You can define tables using emxTable.jsp to present information on tabular format.

In this section:

- [About Tables](#)
- [Building a Table](#)
- [Editable Tables](#)
- [Parameters for Table Objects](#)
- [Refreshing the Table After Adding, Editing, or Deleting Objects](#)
- [Object or Relationship ID of a Selected Table Item](#)
- [Table Columns](#)
- [Additional URL Parameters for Tables](#)
- [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#)
- [JPO Guidelines for Tables](#)
- [Custom Filter for a Table](#)
- [Sorting Programs for Tables](#)
- [Parameters and Settings for Inquiry Objects](#)

About Tables

You can define tables using emxTable.jsp to present information on tabular format. This section provides background details about tables.

The following topics are discussed:

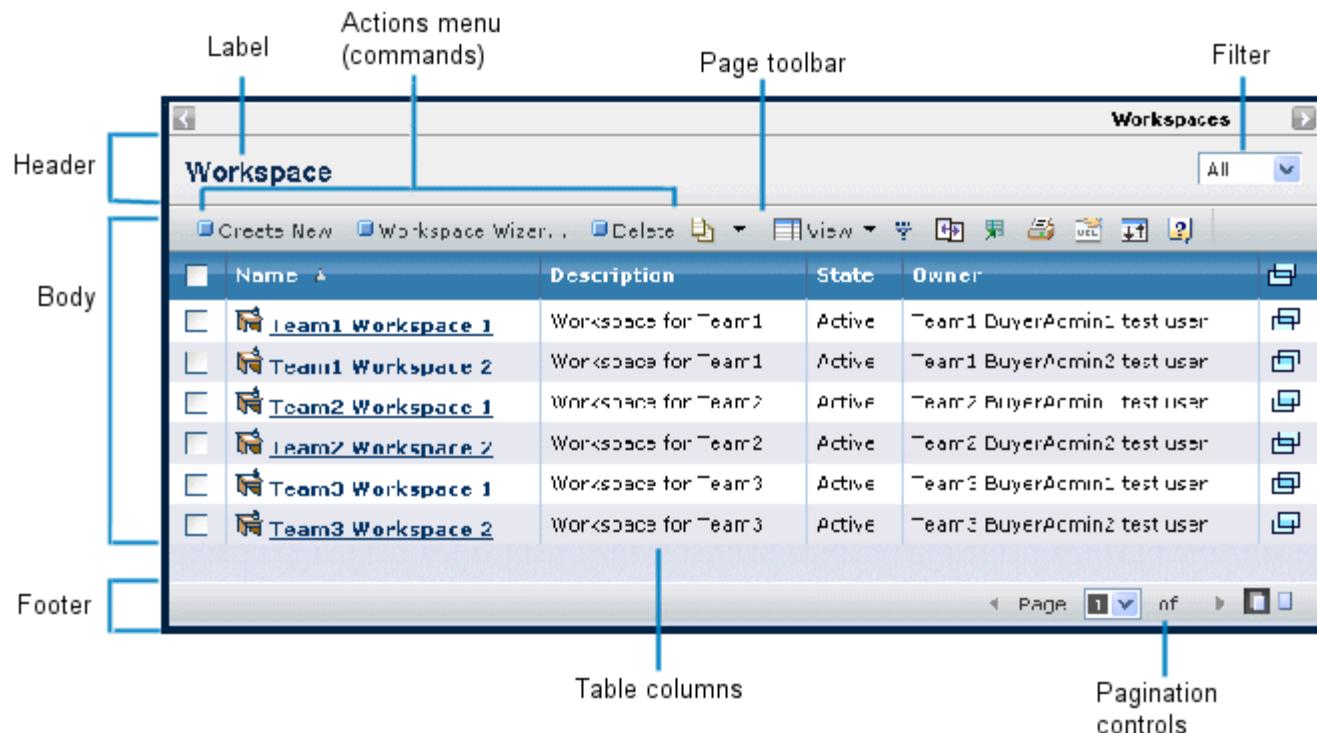
- [Table Components](#)
- [Table Body](#)
- [Table Pagination Controls](#)
- [Object Specific or Non-object Specific](#)
- [Methods for Filtering a Table](#)
- [Mechanisms that Define a Table Page](#)
- [Printer Friendly Mode](#)
- [Column Settings for the Column JPO](#)
- [Table Pages are Linked to an Application](#)

Table Components

Table pages contain three main sections: the header, body, and footer. The primary component of a table page is the table, which is a list of business object.

Table pages also contain other elements, such as a heading, Actions menu, and pagination controls. For example, the graphic below shows a table page for a list of workspaces. Each row represents one business object (or relationship) and each column represents a particular attribute or other characteristic about each business object/relationship.

Once you have created a table, referred to as a system table, you can derive tables from it, called user tables. You can only derive a user table from a system table in MQL. Once the derived user table is created, you can see it in ENOVIA Live Collaboration the same as your other tables. See the *MQL Guide* for details. If you add a column to a system table, it will be propagated to all tables derived from that table.



In this example, the Actions menu shows as a series of commands on the page toolbar. Whether the Actions menu shows or the individual commands show depends on the value of the emxFramework.Toolbar.PivotCommand.Limit property in emxSystem.properties. If the number of commands is over that value, then the Actions menu shows; otherwise, the individual commands show on the toolbar. Of course, the developer may have added individual commands directly to the toolbar instead of the Actions menu.

Table pages display most often in view mode. When users want to print or edit the page, they can click an icon in the table page toolbar to display the table in either printer-friendly mode or, if the table is configured for it and they have appropriate access, edit mode. You can customize table pages to be shown directly in edit or printer-friendly mode. See [Invoking a Table Directly in Edit Mode](#) or [Printer Friendly Mode](#).

Table Body

The table body is an html FORM with the name "emxTableForm". This FORM contains the list of objects to display. The table can be configured to display check boxes or radio buttons. The check box or radio button control is named emxTableRowId.

The checkbox/radio buttons are assigned with the values of ObjectId and RelId, if available. If relId is available, the values are separated by the pipe symbol (|).

```
emxTableRowId = <objectId>
```

or

```
emxTableRowId = <relId>|<objectId>
```

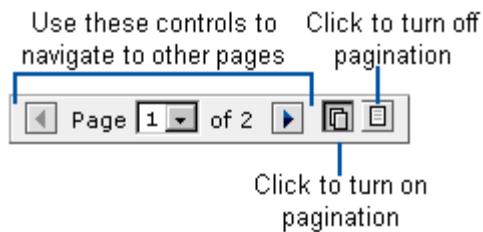


Table Pagination Controls

Paginating a table means the table shows a certain number of rows per page. If the number of rows in the table exceeds that number, the user must navigate to the next page to see the additional rows.

You can configure the number of rows per page by passing the pagination parameter to emxTable.jsp. You can also use the pagination parameter to turn pagination off.

If you do not pass the pagination parameter to emxTable.jsp, the system uses the pagination number specified for the emxFramework.PaginationRange property in emxSystem.properties. By default this property creates a new table page for every 10 rows. You can change this number using the Pagination option in the Preferences page.



Users can turn on and off pagination using the Paginate and Don't Paginate buttons on the right side of the Pagination control panel. When pagination is turned off, all rows are listed on the first table page, and the user must scroll to see the rows. The system removes the navigation controls because there is only one page.

When you configure a table page so that it is paginated and users are allowed to select one or more rows, the system remembers the user's selections across pages. For example, suppose a table page lists parts for a company and there are 18 parts, 10 on the first page and 8 on the second. The user can check a part on the first page and a part on the second page and then click on a toolbar item to process both parts, such as a Delete or report action.

You can turn off the Remembering selections across pages globally using the emxNavigator.UITable.Pagination.RememberSelection property in the emxSystem.properties file. You can turn it on and off per table page using the rememberSelection parameter for emxTable.jsp.

When you configure a toolbar item to process selected rows on a table page, you can show the user how many rows are selected using the macro \${TABLE_SELECTED_COUNT}. For example, in a toolbar item command object, you can use the macro in the Confirm Message setting to display a message such as "You are about to delete \${TABLE_SELECTED_COUNT} parts. Click OK to continue?"



Object Specific or Non-object Specific

Users typically access a table page from a feature listed in a submenu or from a tree category. How the user accesses the table affects some of the behavior for the page. For example, users may access a Routes table page by clicking My Enovia >

 Routes from the (My Desk) menu.

Users can also access a page with the Routes table by selecting **Categories > Routes** from a context object's tree. The table is usually the same as the non-object specific page, but it might have a different heading or different actions. For example, when the Route category displays for a specific Part, that page lists only the Routes for that Part.

Regardless of how users access a table page, clicking the name of an object (or relationship) in the table opens the tree and the Properties page for the object. For example, clicking a feature name from the Features table page brings up the tree for the feature. Trees are implemented as the Categories menu.

The screenshot shows two windows. The top window is titled 'Features' and lists three items: F1, T1, and L1. The item F1 is selected and highlighted with a blue border. A blue arrow points from the 'Name' column of F1 down to the 'Feature F1 rev A: Properties' window below. The bottom window is titled 'Feature F1 rev A: Properties' and displays a sidebar with categories like Assignees, Configuration Rules, and Resources, along with a main pane for F1 A.

Methods for Filtering a Table

There are several ways to configure a table page to let users filter table data. You can use the filter drop-down, the filter tool, or a custom filter.

The screenshot shows a table titled 'Invoices for: Acme Widget'. At the top, there are three filtering methods: 'Custom Filter', 'Filter Table tool', and 'Filter drop-down'. The 'Custom Filter' section includes dropdown menus for 'Type' (set to 'All Listed') and 'Relationship' (set to 'All Listed'). The 'Filter Table tool' section includes dropdowns for 'to' and 'from' fields, and a 'Refresh' button. The 'Filter drop-down' section shows a dropdown menu labeled 'Customer Table' with a list of names. Below the filtering tools is a table with columns: Name, Cust-ID, Street/Ad, City, State, Zip-Code, Country, and Status.

- Filter drop-down--Use the filter drop-down to predefine several lists of objects so users can select the list they want. Use the filter drop-down for filtering that is not based on the column data. For example, the filter list on the RFQs page lets users select a list of all RFQs they own or a list of all RFQs they receive in a route. You configure a table page to show a filter list by passing inquiry object names or a JPO.
- Filter tool--Use the Filter tool to define the columns users can filter the table by. Set Auto Filter=true on a table column

to let users filter by the column. Basic properties or attributes defined by the `emxFramework.AutoFilter.Basic.InclusionList` and `emxFramework.AutoFilter.Attribute.InclusionList` properties in `emxSystem.properties` also use the Filter tool. For example, to let users filter the table by state, set `AutoFilter=true` for the state column. Users can filter the table so it only shows objects in specific states. The `autoFilter` URL parameter overrides any columns with the Auto Filter setting. That is, if a column has `Auto Filter=true` set, but the URL parameter `autoFilter=false` is passed, then the filter tool does not display on that table page.

- Custom filter--Use a custom filter to place filter controls between the table page header and body frames. Implement the custom filter using a custom JSP and by passing the `FilterFramePage` and `FilterFrameSize` parameters to `emxTable.jsp`. For details on implementing the custom filter, see [Custom Filter for a Table](#).

You cannot filter a table based on a column that could contain multiple values in a cell.



Mechanisms that Define a Table Page

The configurable `emxTable.jsp` accepts many parameters that define the content and behavior of the table page. You pass these parameters to the JSP through the href URL.

The main components of the table page--the table and its columns, the query for getting the list of business objects for the table, and the menus--are defined as administrative objects. You pass the names of these objects to the `emxTable.jsp` through its URL parameters.

Each unique set of table columns needs a table administrative object. For example, suppose certain ECR pages need to show the name, revision, and state for ECRs but other pages need to show the name, revision, owner, and description. You would need a table administrative object for each of these tables, such as `ENCECRBasic` and `ENCECROwner`.

You define the rows in a table using inquiry or program administrative objects. By populating a table page with different inquiries/programs, you can get different data with the same table. For example, you can use an ECR's table on one page to show a list of ECRs owned by the user, and you can use the same table on another page to show a list of ECRs for a particular part.



Printer Friendly Mode

You can show a table directly in printer friendly format, without first showing the view mode table, by passing the `style` parameter into `emxTable.jsp`. The page display the same way as when users click the printer friendly button in the table toolbar. The printer friendly page uses the `emxUIListPF.css` style sheet by default. This page display is not configurable using the property in `emxSystem.properties`. See [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#).



Column Settings for the Column JPO

The settings of the configurable table column contain the information describing how to display the column, where the column gets its value, and how to update it. Pass the `columnMap` (a key-value pair the system generates containing all the information about the column) to the JPO methods used with settings `program`, `Update Program` and `Range Program`.

The following table describes the sample structure of the `columnMap`.

Key	Description	Possible or Sample Values
<code>expression_businessobject</code>	An expression that gets the column value from the database.	<code>relationship[EC Distribution List].to.name</code>
<code>Name</code>	Name of the column.	<code>ReviewerList</code>
<code>Label</code>	The column label.	<code>emxComponents.Table.Label.ReviewerList</code>
<code>Href</code>	Href given to the column.	<code>url</code>
<code>Settings</code>	Map of all the column settings with the following key-value pairs and the description: Editable -- true --Whether the column can be edited column Type -- <code>programHTMLOutput</code> -- Determines the column value retrieval procedure Registered Suite -- Components --Application Suite Name	Settings map contains the information related to column settings.

The following sample code shows how to read the field settings of the individual columns from the `columnMap`.

```
// Getting the columnMap
HashMap programMap = (HashMap) JPO.unpackArgs(args);
HashMap columnMap = (HashMap) programMap.get("columnMap");
```

```
// Getting the first-level key-value pair from columnMap
String field_expression = (String)
columnMap.get("expression_businessobject");
String fieldName = (String) columnMap.get("name");

//Getting the Settings key-value pair from columnMap which is
inside nested Hashmap
HashMap settingsMap = (HashMap) columnMap.get("settings");
String updateProgram = (String) settingsMap.get("Update
Program");
String columnType = (String) settingsMap.get("Column Type");
```



Table Pages are Linked to an Application

Users typically access table pages by selecting a link from a submenu or tree category. The component that calls the table page must specify the configurable JSP for table pages, emxTable.jsp, in the href parameter. The href should include any parameters needed to configure the page, as described in [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#).

The following example shows the URL used to create a table page for the My Tasks page:

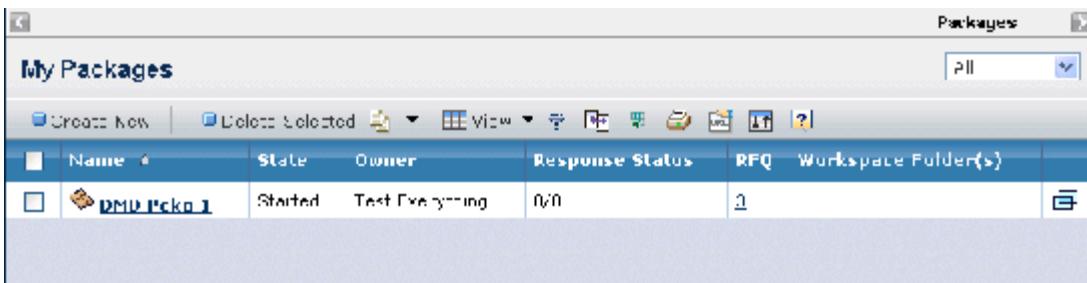
```
 ${COMMON_DIR}/emxTable.jsp?program=emxInboxTask:getMyDeskTasks,
emxInboxTask:getActiveTasks,emxInboxTask:getCompletedTasks,emxInboxTask:getTasksToBeAccepted
&table=APPTaskSummary&programLabel=emxComponents.Filter.AllTasks,emxComponents.Filter.Active,
emxComponents.Filter.Complete,emxComponents.Filter.TasksToBeAccepted&header=
emxComponents.Common.Tasks&toolbar=APPTaskSummaryToolBar
&sortColumnName=Name&sortDirection=ascending&selection=multiple&HelpMarker=emxhelptasks
&selectedFilter=emxInboxTask:getActiveTasks
```

Building a Table

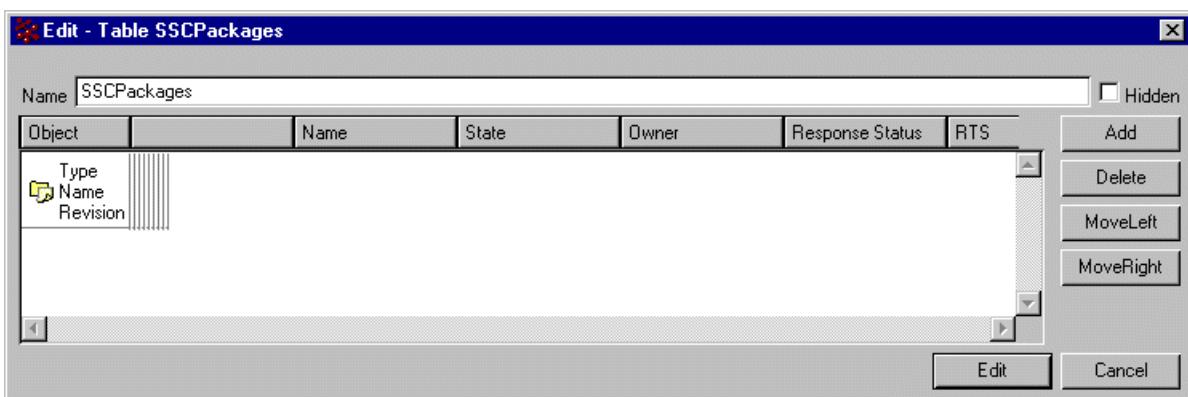
The following procedure lists the main steps for creating a table page and connecting it to an application.

1. Create a table object and define the columns for it.

The graphics illustrate the components to create a table page for Packages. It is called by selecting  > Sourcing > Packages.

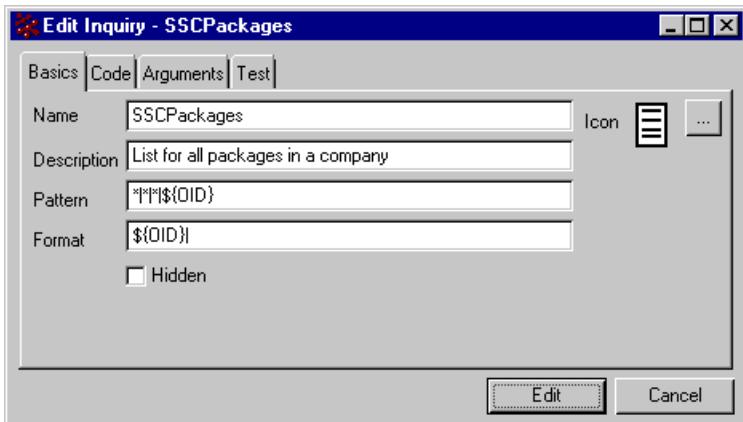
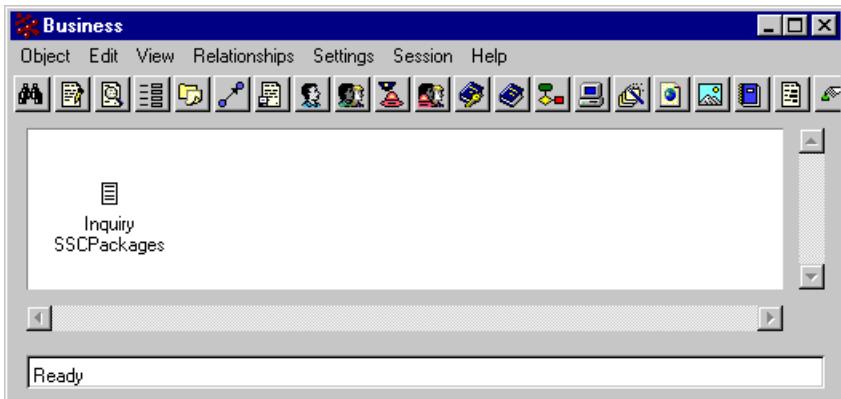


Define the column data, the column heading, the column order, and who can access each column. For a description of how to fill in the parameters and settings for tables and columns, see [Parameters for Table Objects](#). For naming conventions, see [Naming Conventions for UI Administrative Objects](#).



2. Create an inquiry object or program object to define how the system gets the list of business objects (and relationships, if appropriate) when the table page first loads.

For example, the Packages page should display all packages for a person's company. For a description of how to fill in parameters for inquiry objects, see [Parameters and Settings for Inquiry Objects](#). For naming conventions, see [Naming Conventions for UI Administrative Objects](#).



3. Create inquiry or program objects for each filter list option that is different from the inquiry/program needed when the page loads.

For example, the filter list for the Package page should contain options for showing only packages that are owned by the user and only packages routed to the user.

4. Make sure to create the menu and command objects for the table page's Actions menus.

For instructions, see [Toolbars](#).

5. Enter emxTable.jsp and specify the parameters needed to display the page in the href parameter for the menu or command object that calls the table page (or in the JSP if a JSP is calling the page). Since emxTable.jsp is in the ematrix/common directory, the first part of the URL should usually be: \${COMMON_DIR}/emxTable.jsp. Here are two example href values. For a description of the parameters, see [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#).

This sample URL calls emxTable.jsp using the program URL parameter (among others):

```

${COMMON_DIR}/emxTable.jsp?table=SCSBuyerDesksSummary&toolbar=SCSBuyerDesksSummaryToolBar
&program=SourcingBuyerDesk:getBuyerDesks,SourcingBuyerDesk:getAssignedBuyerDesks&programLabel=
emxSourcing.Common.All,emxSourcing.Common.Assigned&header=emxSourcing.BuyerSearchResult.BuyerDesks&HelpMarker=em:
&sortColumnName=Name&sortDirection=ascending

```

6. If you are working with the Web-based user interface as you are making changes and want to see your changes in the user

interface, select > Utilities > Reload Cache and click the browser Refresh button.

The cache refreshes automatically when the component age expires. This setting is in emxSystem.properties.

Only users assigned to the Administration Manager role have access to the Reload Cache tool.

Editable Tables

You can define a table in which users can edit cell values.

In this section:

- [Editable Table Configuration](#)
- [Invoking a Table Directly in Edit Mode](#)
- [Toolbars for an Editable Table](#)
- [Using Type Ahead in a Table Column](#)
- [Manual Entry for Combo Boxes](#)
- [Edit Access for Selected Table Objects](#)
- [Dynamic URLs and mxLinks for a Column](#)
- [Adding an Editable Table to an Application](#)
- [Pre/Post/Cancel Processing of an Editable Table](#)

Editable Table Configuration

You can configure tables to display in editable mode.

You can invoke the editable table directly (see [Invoking a Table Directly in Edit Mode](#)), or you can call it from the view mode of the table. For example, you can add an action link to the view mode of the SCOs summary page to enable the user to open the same table in edit mode.

The edit mode of the table can contain two types of toolbars:

- a configurable toolbar
- a mass update toolbar, that allows changes to multiple rows with a single action

The editable table can contain both toolbars, one toolbar, or neither.

You define the table, columns and configurable toolbar using the same table administrative objects that define view-only tables. Similar to form fields, when defining a table column that should be editable, you need to specify additional settings to tell the system how to handle edits to the column. For example, you need to specify the type of input control, any special validation for the data, and any update program for it.

The configurable JSP, emxTableEdit.jsp, creates editable table pages. The system calls it automatically when the user clicks the Edit link on a view-only table page. The Edit link is included on the view-only table page whenever editLink=true is passed to emxTable.jsp.

You can configure the edit link called from the view mode of a table by:

- Passing a URL parameter called editLink=true. This approach does not support access control.
- Configuring the toolbar command shown in the table with the href assigned to emxTableEdit.jsp. Use this approach when you need access control for the edit command. You must also configure these associated settings:
 - Submit=true

If you want the edit page to open in a separate window, also include these parameters:

- Popup Modal=true
- Target Location=popup

For example, this graphic shows a standard view-only table that lists Resources. When the user clicks the Edit All link...

	Name	Min	Max	Initial	Comments	Mandatory	Inherited	Design Responsibility	Owner	Inherited From
	RES-000001	5.0	7.0	6.7	Software Developer	Yes	False	MyCompany	Engineering Test	

...the same table displays in edit mode, but some column values are editable, as shown below.

Name	Min	Max	Initial	Comments	Mandatory	Inherit
RES-000001	7.0	8.7	6.0	Software Developer	Yes	False

Like form pages displayed in edit mode, you can configure which columns can be edited. In the example shown above, users cannot edit the Name column, but can edit the Min, Max, Initial, Comments, and other columns.

After users change the data, they save it by clicking Done. If the column type is not businessobject or relationship, you must write an update program and update function and specify it to update the table data.

The editable table does not support filters. The currently-selected filter is displayed (if one is defined), but users cannot

change the selection. The editable table also does not support pagination controls. It displays what is shown in the view-only page.

Invoking a Table Directly in Edit Mode

You need to provide parameters to view a table in edit mode.

The `table` and one of `program` or `inquiry` are required parameters for editable tables. When the editable table page is launched through a view table, the editable table gets these parameters internally.

You can display the editable table page directly by calling `emxTableEdit.jsp` and passing the required URL parameters table and one of program or inquiry and optional parameters listed in [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#). You can directly invoke the editable table to show in any frame or in a separate pop-up dialog. If it displays in any frame other than in a pop-up dialog, the Cancel button does not display. When the user clicks the Done button, the editable table page refreshes.

The following shows a sample URL used to directly invoke an editable table:

```
 ${COMMON_DIR}/emxTableEdit.jsp?HelpMarker=emxhelpresourceusageedit
 &program=emxFixedResource:getAllFixedResourcesForEditAll&table=FTRFixedResourceList
 &timeStamp=null&header>Edit:Resource Rule
```

Toolbars for an Editable Table

You can define the editable table page to include a configurable toolbar and/or a mass update toolbar.

The edit mode of the table can contain two types of toolbars:

- Mass update toolbar allows changes to multiple rows with a single action. Set the emxFrameworkShowMassUpdate property to true to include it. You can adjust this at the page level by passing the massUpdate=false URL parameter to emxTableEdit.jsp. When set to true, the editable table contains check boxes before each row of the table, and it also displays a mass update toolbar at the top of the table. Users can select the column and values to change for the rows selected.
- Configurable toolbar can be the same as the toolbar used in view-only mode. You can include a configurable toolbar in the header of an editable table when the toolbar name is passed to emxTable.jsp or emxTableEdit.jsp using the editToolbar parameter. You can include the same toolbar in both the view-only mode and the Edit mode, as long as it does not contain commands that would change or refresh the view-only table or editable table. If you show the Edit button in the Table component by passing the parameter editLink = true to emxTable.jsp, then you should pass the editToolbar parameter to emxTable.jsp.

If the Edit button in the Table component is a custom command, where the URL is a custom URL (for example emxTableEdit.jsp), you should append the parameter to the custom URL (emxTableEdit.jsp).

You can include the generic delete command, [AEFGenericDelete](#), as part of a toolbar or Actions menu in a toolbar. This command provides delete functionality for the table.

The table can contain both toolbars, either toolbar, a plain toolbar with just a help icon, or no toolbar at all (by passing the parameter HelpMarker = false).

The export and printer-friendly icons are not shown in the toolbar of an editable table even if they are configured to show in the view-only table.

Using Type Ahead in a Table Column

When editing data in a table column, the column can be configured with the type ahead feature to allow users to quickly enter the needed data.

Type ahead can be achieved using any of these methods:

- Advanced Search that uses emxFullSearch.jsp to retrieve possible values for a column cell. See [Advanced Search Used for Type Ahead](#).
- Pre-defined Choosers for Person, Organization, or Type columns. See [Pre-Defined Type Ahead Choosers](#).

Manual Entry for Combo Boxes

You can configure combo box fields in web forms or editable tables to allow users to type a value into a text box.

When users select the ~Add Manually~ option from the combo box, or a range value defined for manual edit, the text box is enabled for manual entry. For example, if a combo box has a list of the colors red, blue, green, yellow, and other, users could select other and then type white in the text box.

No validation is done on the field, so any string can be entered, even if it does not relate to the field attribute.

The following shows a combo box for a Color field, with the adjacent text box enabled for manual edit. In this case, the selected option that allows manual entry is " ~ Add Manually ~".



In the above example, the range value for manual entry is added as "!=?~ Add Manually ~". The "!=" indicates to the system that selection of this particular range value must enable the manual entry text box. You can only define one range value for a given attribute in this way.

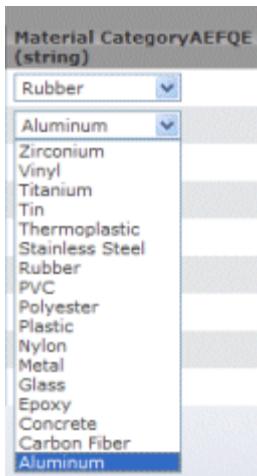
In conjunction with the range value definition, you must use the following field settings:

```
Allow Manual Edit = true  
Editable = True  
Input Type = combobox
```

Optionally, you can specify the sort direction for the combo box as ascending (default), descending, or none:

```
Sort Direction = ascending
```

If you set Allow Manual Edit to false, the combo box is shown alone without an adjacent text box, and if there is a range value defined that is preceded with "!=" , it is excluded from the list of options in the combo box. The following shows a combo box with Allow Manual Edit set to false:



Edit Access for Selected Table Objects

When viewing objects within a table, the user can select which objects to edit. For example, if the user is viewing page 5 of a table and wants to edit only three of the objects on that page, the user can select those three and click Edit to view an editable table containing only those selected objects.

You can control the access of each cell or a complete row of the Table Edit page.

The following topics are discussed:

- [Row Edit Access by the objectList](#)
- [Cell Edit Access](#)

Row Edit Access by the objectList

You can control row edit access by the program JPO that is configured to fetch the list of objects for the table by adding a new key to each HashMap in the MapList given by the JPO program.

The new HashMaps of Maplist should contain:

- id
- id[connection]
- RowEditable

The following table contains the list of keys with possible values:

Key	Value
id	Objectid of the object
id[connection]	Relationship id of the object, if any
RowEditable	The value can be any of the following: show --The row is editable and those columns with the setting "Editable = true" display as editable. This is the default value. hide --This row will not display in the TableEdit page (cannot be used with Structure Browser tables). readonly --The row displays as read only (non-editable) even if the columns have the setting "Editable = true".

Sample Code

Here is the Sample Code used in one of the applications:

```
MapList listBos = new MapList();
listBos = connectedDoc.getRelatedObjects(context,
RELATIONSHIP_CHARACTERISTIC, characteristicType, busSelects, null, false,      true,
(short) 1, null, null);
String characLevel = "";    String characClaimed = "";
if(listBos != null) {
    for(int i=0; i < listBos.size(); i++)  {
        Map characMap = (Map)listBos.get(i);
        characLevel = (String)characMap.get("attribute[" + attrLevel + "]");
        characClaimed = (String)characMap.get("attribute[" + attrClaimed + "]");
        if ("True".equalsIgnoreCase(isTemplate) &&
!characLevel.equalsIgnoreCase(templateLevel)) {
            characMap.put("RowEditable","readonly");
        } else if ("False".equalsIgnoreCase(isTemplate) &&
"Yes".equalsIgnoreCase(characClaimed)) {
            characMap.put("RowEditable","readonly");
        }
    }
}
return listBos;
```



Cell Edit Access

You can control the cell level edit access by setting "Edit Access Mask" on the table column. You can further filter the edit access to the cell level by using a filter access expression or by a filter access JPO defined on the access of the object's

policy.

Column Setting	Accepted Values
Edit Access Mask	Any access mask, for example, modify, connect

The system does the access check on all objects in one database call when it retrieves the column values. Cells that do not have the specified access display as read-only. For example, Specification Central's Micro Biological Characteristics type objects cannot be editable if the object's attribute Claimed value is Yes. You should also set the following:

- For all the columns with Editable = true, set Row Edit Mask=modify
- In the Characteristic policy for some users, establish a filter attribute[Claimed] ==Yes in all states
- When the filter expression evaluates to true, then that user gets modify, read and show access in the appropriate states of the policy

The objects with the Claimed attribute set to Yes display as read-only in the TableEdit component.

Dynamic URLs and mxLinks for a Column

When a column has the Dynamic URL setting set to enable, users can enter URLs or mxLinks in an editable table, and the system displays those values as hyperlinks when in view mode.

The valid URL formats are defined by the `emxFramework.DynamicURL.Keywords` parameter in the `emxSystem.properties` file.

The `emxFramework.DynamicURLEnabled` system parameter controls this feature. If enabled at the system level (in the `emxSystem.properties` file), it can be disabled for specific columns by passing the `Dynamic URL = disable` setting.

Dynamic URLs can be entered in any free-form text fields. Dynamic URLs *cannot* be entered in a field if:

- The Dynamic URL setting is set to disable (entered URL value will display as text)
- The Column Type setting is set to programHTMLOutput
- The Format setting is set to date or numeric
- Configured with businessobject or relationship expressions as attributes with datatypes Boolean, date/time, real, integer or associated with a dimension
- Configured with businessobject or relationship expressions as attributes with predefined fixed range values
- Configured with businessobject basic expressions as attributes (such as Type, Name, Rev, Vault, Owner) except Description

Users enter the URL in straight text, such as "`http:\\server.com\abc.jsp?param=value`" or `www.enovia.com`, and then the system converts the text to hyperlinks. These examples would be converted to `http:\\server.com\abc.jsp?param=value` and `www.enovia.com`, respectively.

When Dynamic URLs are enabled, mxLink values are also enabled. Users can enter an mxLink value in a text field, and the framework converts that entry into a hyperlink. The user can type in the mxLink keyword and value, or search for the mxlink value. See the *Application Exchange Framework User's Guide* for details on how a user enters the value.

The format entered in the text field is:

`mxlink:<type>|<name>|<revision>|<vault>`

For example:

`mxLink:Part|Part-5000-01|0|eService Production`

This string is translated to a URL pointing to the default page for the object, such as:

`http://server:8080/ematrix/common/
emxNavigator.jsp?objectId=32902.4369.56921.33541`

and displays as a hyperlink such as:

`Part Part-5000-01 0`

When the user clicks exits an editable field, the table validates the mxLink entries and displays an error message if any of the values could not be resolved. The user has the option of saving the data anyway, or canceling and fixing the error. If more than 10 errors are found, the message lists up to 10 errors and indicates there are additional errors.

When processing the `<type>` of an mxLink, the framework uses the symbolic name (for example, `type_Part`), and looks up that symbolic name in the string resources properties file for the language the browser is set to and displays the internationalized name of the type. If the string resources properties file does not contain an entry for the type's symbolic name, then the type displays as entered.

See the *Application Exchange Framework User's Guide* for variations on this format, such as omitting vault or revision, and using the latest revision identifier. If a user omits the vault or in the mxLink, the framework automatically adds the vault to the stored hyperlink. If a user omits the revision identifier, the framework automatically adds the latest revision identifier to the stored hyperlink.

In edit mode, if the column containing a dynamic URL or mxLink is defined as non-editable, then it shows as a link the same as in view mode.

Adding an Editable Table to an Application

You add an editable version of a table page by adding the `editLink=True` parameter to the URL that calls the table.

If not passed, the system assumes the parameter is false. When `editLink=true` is included, the table page includes an Edit link in the toolbar or Actions menu. This link is configured to call `emxTableEdit.jsp`. The system automatically passes the `suiteKey` and `header` parameters to the jsp, using the same values defined for the `emxTable.jsp`. This means the title for the editable version of the table is the same as the title for the view-only version.

In addition, make sure all columns that should be editable include the `Editable=true` setting and other settings that define the type of input control, validation, and update programs needed to edit the data. For a list of parameters and settings, see [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#). For examples of different types of editable columns, see [Table Column Data Definition](#).

Write any programs needed to get data or update data and create program objects to represent them.

Pre/Post/Cancel Processing of an Editable Table

You can define a URL or a JPO to executed before loading an editable table page, whenever a user clicks a Cancel button or closes the editable table component, or after completion of the edit process.

In this section:

- [Pre/Post/Cancel Processing of an Editable Table](#)
- [Preprocess URL for an Editable Table](#)
- [Preprocess JPO for an Editable Table](#)
- [Cancel Process URL for an Editable Table](#)
- [Cancel Process JPO for an Editable Table](#)
- [Post Process URL for an Editable Table](#)
- [Post Process JPO for an Editable Table](#)

Pre/Post/Cancel Processing of an Editable Table

The system invokes pre processing before loading the editable table page. The system invokes cancel processing whenever a user clicks a Cancel button or closes the editable table component. The system invokes post processing after completion of the edit process. The edit process, post process JSP, and post process JPO are performed in a single transaction. If an exception occurs during the edit process or post process, the system rolls back the entire transaction.

You can specify a pre/post/cancel processing JSP or JPO by passing the parameters listed in the table below to the appropriate JSP files:

- emxTable.jsp
- emxTableEdit.jsp (for direct edit modes)

These parameters are applicable only for edit mode.

URL Parameter	Possible/Default Values	Description
preProcessJPO	<JPOName>:<MethodName>	The JPO to use when a user clicks Edit on a table component.
preProcessURL	\${SUITE_DIR}/emxCustomPreProcess.jsp or ..<ApplicationDirectory>/emxCustomPreProcess.jsp Example: ../engineeringcentral/emxCustomPreProcess.jsp	The JSP called before displaying an editable table. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.
postProcessJPO	<JPOName>:<MethodName>	The JPO to execute when a user clicks the Done button on a table component.
postProcessURL	\${SUITE_DIR}/emxCustomPostProcess.jsp or ..<ApplicationDirectory>/emxCustomPostProcess.jsp Example: ../engineeringcentral/emxCustomPostProcess.jsp	The JSP called when a user clicks the Done button on a table page. The system calls the configured JSP after edit processing and database update. Specify the JSP name using the macro for the page location, or use the relative path shown in the examples.
cancelProcessJPO	<JPOName>:<MethodName>	The JPO to execute when a user clicks the Cancel or close window button on a table component.
cancelProcessURL	\${SUITE_DIR}/emxCustomCancelProcess.jsp or ..<ApplicationDirectory>/emxCustomCancelProcess.jsp Example: ../engineeringcentral/emxCustomCancelProcess.jsp	The JSP called when a user clicks the Cancel button or closes the table page. Specify the JSP name using the macro for the page location, or use the relative path shown in the examples.

Preprocess URL for an Editable Table

The preProcessURL parameter specifies the name of the JSP to call during pre-processing of the table in edit mode.

The following examples show how to invoke the emxTable.jsp with preProcessURL:

1. Set the href of any command object using macros as appropriate:

```
 ${COMMON_DIR}/emxTable.jsp?mode>Edit&table=<table_name>
 &preProcessURL=${SUITE_DIR}/emxCustonPreProcess.jsp
```

or

2. Invoke emxTable.jsp with custom JSP pages using the relative path, (macros are not supported):

```
 ./common/emxTable.jsp?mode>Edit&table=<table_name>
 &preProcessURL=../<Application Directory>/emxCustonPreProcess.jsp
```

If both a preProcessURL and preProcessJPO are specified, the preProcessURL executes first followed by the preProcessJPO.

The pre process JSP supports the following input parameters:

- The request parameters available to the emxTable.jsp, for example, `timeStamp`, `portalMode`, `suiteKey`, etc.
- The request parameter `timeStamp` gets the table data stored in the table bean as a map.
- To update or interact with any objects, obtain the context from the request like this:

```
context = (matrix.db.Context)request.getAttribute("context");
```

To manipulate only those objects selected before entering edit mode of the table, the JPO method should use the `EditObjectList` MapList in the `tableData` map. This MapList contains the objectIDs of the selected objects, while the `ObjectList` MapList contains the objectIDs of all objects in the table.

The following code sample shows how to read custom processing page values from the `requestMap`:

```
<%
    // get the timeStamp from the incoming HttpServletRequest
    String timeStamp = (String) emxGetParameter(request, "timeStamp");

    // define the table bean
%>
<jsp:useBean id="tableBean"
class="com.matrixone.apps.framework.ui.UTable" scope="session"/>
<%
    // get the tableDataMap and the requestMap from the table bean
    HashMap tableDataMap = (HashMap) tableBean.getTableData(timeStamp);
    HashMap requestMap = (HashMap)
    tableBean.getRequestMap(tableDataMap);

    // get specific values from the requestMap
    String languageStr = (String) requestMap.get("languageStr");
    String timeZone = (String) requestMap.get("timeZone");
    String charSet = (String) requestMap.get("charSet");
    String suiteKey = (String) requestMap.get("suiteKey");
    String stringResourceFile = (String)
    requestMap.get("StringResourceFileId");
%>
```

This code sample shows how a custom processing page can iterate through the table objectIDs. This example 'reserves' all objects in the `objectList` (typically done in a pre processing custom JSP page). Note that custom processing JSP pages do not return anything to the component.

```
<%
    // get the timeStamp from the incoming HttpServletRequest
    String timeStamp = (String) emxGetParameter(request, "timeStamp");

    // define the table bean
%>
<jsp:useBean id="tableBean"
class="com.matrixone.apps.framework.ui.UTable" scope="session"/>
<%
    // get the tableDataMap and the objectList from the table bean
    HashMap tableDataMap = (HashMap) tableBean.getTableData(timeStamp);
    MapList objectList = (MapList)
    tableBean.getObjectList(tableDataMap);

    // loop through the objectList's list of object Maps
    Iterator objectListItr = objectList.iterator();
    while(objectListItr.hasNext()){

        // get the current object Map
        Map curObjectMap = (Map) objectListItr.next();
        // get the current object's objectId from the current object Map
        String curObjectId = (String) curObjectMap.get("id");
        // a comment to pass along with the object reserve operation
        String reserveComment = "This object is Reserved";
    }
%>
```

```
// create a BusinessObject, open the object, reserve the object,
close the object
BusinessObject curBusObject = new BusinessObject(curObjectId);
curBusObject.open(context);
    curBusObject.reserve(context,reserveComment);
curBusObject.close(context);

%> }
```

Preprocess JPO for an Editable Table

The preProcessJPO parameter specifies the name of the JPO program and method to invoke during pre processing.

This example shows how to invoke the emxTable.jsp with preProcessJPO:

```
 ${COMMON_DIR}/emxTable.jsp?mode=Edit&table=<table_name>
 &preProcessJPO=<JPO Name>:<Method Name>
```

If both a preProcessURL and preProcessJPO are specified, the preProcessURL executes first followed by the preProcessJPO.

The pre process JPO specifies the programMap as an argument for the pre processing. The programMap contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains these key parameters and expected values:

Parameter	Description
objectId	Object ID
relId	Relationship ID of the object

- `tableData` contains all the table information

The above maps are packed using the packArgs method supported by JPO and passed to the pre process JPO being invoked. The pre process JPO can unpack this input parameter and use it for custom coding.

If you want to manipulate only those objects selected before entering edit mode of the table, the JPO method should use the EditObjectList MapList in the tableData map. This MapList contains the objectIDs of the selected objects, while the ObjectList MapList contains the objectIDs of all objects in the table.

To display specific objects in the table as read-only, the individual ObjectMap within the ObjectList MapList must have `RowEditable=readonly`. This applies only to the ObjectList that the preprocessing method passes back to the table.

The RowEditable setting controls the entire row as editable or not. The row can hold data on a specific business object or relationship instance, or another set of related values. These values can come from a program which may have dependencies. Currently, the preProcess JPO does not have an interface to control which type of data is editable or non-editable within a row.

You can read the arguments passed for the pre process JPO as shown below:

```
HashMap programMap = (HashMap) JPO.unpackArgs(args);
HashMap paramMap = (HashMap) programMap.get("paramMap");
HashMap requestMap = (HashMap) programMap.get("requestMap");
HashMap tableMap = (HashMap) programMap.get("tableMap");
```

The requestMap contains the languageStr. You can read the value as shown below:

```
String languageStr = requestMap.get("languageStr");
```

The JPO can use the same method as described in [Preprocess URL for an Editable Table](#), to get the objectList, or it can obtain them from the programMap that the table component passes to the JPO. The following code sample shows how a custom processing JPO can iterate through the table objectIds. This example reserves all of the objects in the objectList (typically done in a preprocessing JPO):

```
// unpack the incoming arguments into a HashMap called 'programMap'
HashMap programMap = (HashMap)JPO.unpackArgs(args);
// get the 'tableData' HashMap from the programMap HashMap

HashMap tableData = (HashMap) programMap.get("tableData");
// get the 'objectList' MapList from the tableData HashMap
MapList objectList = (MapList) tableData.get("ObjectList");
// loop through the objectList's list of object Maps
Iterator objectListItr = objectList.iterator();
while(objectListItr.hasNext()){

    // get the current object Map
    Map curObjectMap = (Map) objectListItr.next();
    // get the current object's objectId from the current object Map
    String curObjectId = (String) curObjectMap.get("id");
    // a comment to pass along with the object reserve operation
    String reserveComment = "This object is Reserved";

    // create a BusinessObject, open the object, reserve the object,
close the object
    BusinessObject curBusObject = new BusinessObject(curObjectId);
    curBusObject.open(context);
    curBusObject.reserve(context,reserveComment);
    curBusObject.close(context);

}
```

The pre process JPO returns a HashMap containing the keys defined in this table.

Key	Value
Action	Assigned to one of these values: CONTINUE - the process continues STOP - the process stops and the edit dialog is closed. Default = CONTINUE.
Message	Either a string resource key or an original text message to be displayed to the user. The string must use proper JavaScript format for escape characters such as single quote, double quote, and new lines.
ObjectList	The MapList of object maps that contains the RowEditable setting value based on the reserve status. See the <i>MQL Guide</i> for details on reserving objects.

If the value of the Action key is CONTINUE, then the standard edit page displays. If the value of the Action key is STOP, then the standard edit page does not display.

If the MESSAGE key contains a value, it displays to the user as an alert message on top of the editable page (which is blank if the Action is Stop). When the user clicks OK in the message dialog, then the editable table page displays (if Action is Continue) or the blank page is closed (if Action is Stop).

When the blank window is closed (when Action=STOP; MESSAGE contains a value), do not call the onUnload event (cancelProcessJPO or cancelProcessURL).

Cancel Process URL for an Editable Table

The `cancelProcessURL` parameter specifies the name of the JSP to call during cancel processing of the table in edit mode.

The following examples show how to invoke the `emxTable.jsp` with `cancelProcessURL`:

1. Set the href of any command object using macros where appropriate:

```
 ${COMMON_DIR}/emxTable.jsp?mode>Edit&table=<table_name>
 &cancelProcessURL=${SU
 ITE_DIR}/emxCustonCancelProcess.jsp
```

or

2. Invoke `emxTable.jsp` with custom JSP pages using the relative path (macros are not supported):

```
 ../common/emxTable.jsp?mode>Edit&table=<table_name>
 &cancelProcessURL=../<Application Directory>/emxCustonCancelProcess.jsp
```

If both a `cancelProcessURL` and `cancelProcessJPO` are specified, the `cancelProcessURL` executes first, followed by the `cancelProcessJPO`.

The cancel process JSP contains these input parameters:

- The request parameters that were available for the `emxTable.jsp`, for example, `timeStamp`, `portalMode`, `suiteKey`, etc.
- The request parameter `timeStamp` gets the table data (as a map) stored in the table bean.
- To update or interact with any objects, obtain the context from the request using this code.

```
context = (matrix.db.Context)request.getAttribute("context");
```

The code sample in [Preprocess URL for an Editable Table](#) shows you how to read values from the `requestMap` and iterate through the `ObjectList` for the `objectIds`.

Cancel Process JPO for an Editable Table

The cancelProcessJPO parameter specifies the name of the JPO program and method to invoke during cancel processing.

Example to invoke the emxForm.jsp with cancelProcessJPO:

```
 ${COMMON_DIR}/emxTable.jsp?mode>Edit&table=<table_name>  
&cancelProcessJPO=<JPO Name>:<Method Name>
```

If both a cancelProcessURL and cancelProcessJPO are specified, the cancelProcessURL executes first, followed by the cancelProcessJPO.

The cancel process JPO specifies the programMap as an argument for the cancel processing. The programMap contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains these key parameters and expected values:

Parameter	Description
objectId	Object ID
relId	Relationship ID of the object

- `tableData` contains all the table information

The above maps are packed using the packArgs method supported by JPO and passed to the cancel process JPO being invoked. The cancel process JPO can unpack this input parameter and use it for custom coding.

You can read the arguments passed for the pre process JPO as given below

```
HashMap programMap = (HashMap) JPO.unpackArgs(args);  
HashMap paramMap = (HashMap) programMap.get("paramMap");  
HashMap requestMap = (HashMap) programMap.get("requestMap");  
HashMap tableMap = (HashMap) programMap.get("tableMap");
```

The requestMap contains the languageStr. You can read the value as shown below:

```
String languageStr = requestMap.get("languageStr");
```

The sample code in [Preprocess JPO for an Editable Table](#) shows how to retrieve the objectIds for the table.

The cancel process JPO returns a HashMap or returns nothing. The HashMap contains a single key, Message, that contains either a string resource key or an original text message to be displayed to the user. If the cancel Process JPO does not need to communicate to the table, it can return nothing (`void`) or an empty HashMap. In this case, the table will unload the page after completing the cancel process.

Post Process URL for an Editable Table

The postProcessURL parameter specifies the name of the JSP to call during post processing of the table in edit mode.

If the table is loaded directly into edit mode using emxTableEdit.jsp, note the following behavior. When a user clicks Done on an editable table page, the custom preprocessing methods are reexecuted because the page is refreshed. As a result, if a post processing method has been defined for the table page that unreserves an object (or objects), the re-execution of the preprocessing method will most likely reserve the object(s) again. For customizations that reserve objects in the preprocessing JPO, use a Cancel processing method to unreserve objects instead of a Post processing method.

The following examples show how to invoke the emxTable.jsp with postProcessURL:

1. Set the href of any command object using macros where appropriate as part of the setting:

```
 ${COMMON_DIR}/emxTable.jsp?mode>Edit&table=<table_name>  
&postProcessURL=${SUITE_DIR}/emxCustomPostProcess.jsp
```

or

2. Invoke emxTable.jsp with custom JSP pages using the relative path (macros are not supported):

```
 ./common/emxTable.jsp?mode>Edit&table=<table_name>  
&postProcessURL=../<Application Directory>/emxCustomPostProcess.jsp
```

If both a postProcessURL and postProcessJPO are specified, the postProcessURL executes first, followed by the postProcessJPO.

When the standard edit process completes, the edit display frame `tableEditDisplay` is submitted to a hidden target frame with the post process URL. The post process JSP has the following input parameters available for post processing:

- The request parameters that were available for the emxTable.jsp, for example, `timeStamp`, `portalMode`, `suiteKey`, etc.
- The request parameter `timeStamp` gets the table data (as a map) that is stored in the table bean.
- All the column data of the configurable edit table displayed in the frame `tableEditDisplay` are available to the post process JSP as request parameters.

To update or interact with any objects, obtain the context from the request using this code:

```
context = (matrix.db.Context)request.getAttribute("context");
```

The code sample in [Preprocess URL for an Editable Table](#) shows you how to read values from the requestMap and iterate through the ObjectList for the objectIds. If you want the post process JSP or JPO to roll back the transaction based on certain error conditions, then it has to raise an exception. The configurable table component catches the exception and rolls back the entire edit and post processing transaction.

Post Process JPO for an Editable Table

The postProcessJPO parameter specifies the name of the JPO program and method to invoke during post processing.

If the table is loaded directly into edit mode using emxTableEdit.jsp, note the following behavior. When a user clicks Done on an editable table page, the custom preprocessing methods are re-executed because the page is refreshed. As a result, if a post processing method has been defined for the table page that unreserves an object (or objects), the re-execution of the preprocessing method will most likely reserve the object(s) again. For customizations that reserve objects in the preprocessing JPO, use a Cancel processing method to unreserve objects instead of a Post processing method.

This example shows how to invoke the emxTable.jsp with postProcessJPO:

```
 ${COMMON_DIR}/emxTable.jsp?mode>Edit&table=<table_name>
 &postProcessJPO=<JPO Name>:<Method Name>
```

If both a postProcessURL and postProcessJPO are specified, the postProcessURL executes first followed by the postProcessJPO.

The post process JPO specifies the programMap as an argument for post processing. The programMap contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains key parameters like objectId, relId, languageStr

Parameter	Description
objectId	Object ID
relId	Relationship ID of the object
EditAction	Assigned to this value: DONE - Default value; edit dialog to be closed

- `tableData` contains all the column information

The above maps are packed using the packArgs method supported by JPO and passed to the post process JPO being invoked. The post process JPO can unpack this input parameter, and it can be used by the post process for custom coding.

You can read the arguments passed for the pre process JPO as given below

```
HashMap programMap = (HashMap) JPO.unpackArgs(args);
HashMap paramMap = (HashMap) programMap.get("paramMap");
HashMap requestMap = (HashMap) programMap.get("requestMap");
HashMap tableMap = (HashMap) programMap.get("tableMap");
```

The requestMap contains the languageStr. You can read these parameter values as shown below:

```
String languageStr = requestMap.get("languageStr");
```

The code sample in [Preprocess JPO for an Editable Table](#) shows you how to read values from the requestMap and iterate through the ObjectList for the objectIds.

The post process JPO returns a HashMap or returns nothing. The HashMap contains the keys defined in this table:

Key	Value
Action	Assigned to one of these values: CONTINUE - the transaction is committed STOP - the transaction is aborted. Default = CONTINUE.
Message	Either a string resource key or an original text message to be displayed to the user. The string must use proper JavaScript format for escape characters such as single quote, double quote, and new lines.

If the Message key contains any value, it displays to the user as an alert.

If the value of the Action key is Continue, the system commits the entire transaction of the edit process, post process JSP, and post process JPO.

If the value of the Action key is Stop, the system aborts the entire transaction, including the edit process.

If the post Process JPO does not need to communicate to the table, it can return nothing (void) or an empty HashMap. In this case, the process proceeds with the default behavior of Continue and No message.

Parameters for Table Objects

This table describes the available parameters for table objects. Except for the table name, a table object is a set of column definitions so these parameters apply to columns within a table.

For specific instructions on how to create table objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Access (user MQL command)	<p>The persons, roles, and groups who can access the column. When you assign a role or group, all child roles/groups also receive access. To make the column available to all users, regardless of role/group assignments, choose All.</p> <p>Note that if no users are assigned access, the system assumes all users have access.</p> <p>Also see User Access to UI Components.</p>	Names of group, role, person administrative objects. or All (default)
Alt	Applicable for table columns only when the column type is set to Icon.	Name of what the icon represents.
Applies To	What the select expression applies to: Business object or relationship. Applying an expression to a relationship gets a table of objects related to a known object. For example, getting all EBOMs for a part.	Business Object Relationship
Expression	<p>The select expression to get the column data. This expression is applied to either the relationship or business object, as specified in the Applies To options.</p> <p>Note that the Expression and Applies to options in Business are equivalent to the businessobject and relationship MQL commands.</p> <p>For more information on specifying column data using an expression, see Column Values as Select Expressions.</p>	For business objects: type name \$<attribute[attribute_Originator].value> For relationships: \$<attribute[attribute_FindNumber].value> \$<attribute[attribute_Qty].value>
Heading (label MQL command)	<p>The text to display as the column header.</p> <p>Either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components.</p> <p>The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text.</p>	emxEngineeringCentral.common.Name emxTeam.Common.ProjectName Description
hidden	Hides or shows a column. If a column is hidden (true), it does not show up in the default system table for an object type.	true false (default)
href	<p>The URL that gets executed when the column data is clicked. When a user clicks the hyperlinked column data, the system passes the objectId parameter as part of the URL. By default, the value for the objectId parameter is the ID of the business object for the current row. Using the Alternate OID expression setting, you can configure the field to pass the ID for a different business object.</p> <p>The value for the href parameter should be a JSP and any associated parameters. You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components.</p>	\${COMMON_DIR}/emxTree.jsp \${SUITE_DIR}/emxpathEditPartDialog.jsp
Name	The name of the column used as identifier for the column within the table object.	Name Revision Originator

Name (of table)	Name of the table administrative object. Use the actual table name; you cannot use symbolic names for this parameter. For naming conventions, see Naming Conventions for UI Administrative Objects .	ENCBOMList
RangeHref (range MQL command)	<p>When configuring a table column in Business Modeler, the RangeHREF parameter is set on the Link tab.</p> <p>Use to configure a textbox that has a Browse (...) button that calls a chooser or custom window from which users can select a value to populate the textbox. This Owner textbox is an example with a range helper.</p>  <p>In the RangeHref parameter, specify the href URL to display the window. For example, the href might call a custom selection page.</p> <p>Range helpers are used for table pages only in Edit mode. The only control that can be configured with a range helper is textbox. To specify the control type, set Input Type=textbox.</p> <p>To see an example of a field configured with a RangeHref, see Field with Popup Range Helper. Also see Implementing Range Helpers for Choosers or Custom Pages.</p>	<p>You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For more information, see Using Macros and Expressions in Dynamic UI Components.</p> <p>For example:</p> <pre> \${COMMON_DIR}/ emxSelectVault.jsp \${SUITE_DIR}/ emxSelectUser.jsp emxTypeChooser.jsp?typeList=type_Part,type_Document ./common/ emxTypeChooser.jsp?&selectType=multiselect &SelectAbstractTypes=true&InclusionList=eServiceEngineeringCentral.Types&observeHidden=true&>ShowIcons=true </pre>
Settings	Additional settings that define the behavior and appearance of the column. See Settings for Table Column Objects .	Name/value pairs.
sorttype	<p>Defines how the column values should be sorted.</p> <p>This parameter currently is not available in Business Modeler and can only be set and modified using the sorttype command in MQL. For example:</p> <pre> add table TestTable1 system column name Type label emxEngineeringCentral.Part.Type user all businessobject type setting "Registered Suite" EngineeringCentral sorttype numeric ; </pre>	<p>numeric--Sorts the column values numerically.</p> <p>alpha (Default)--Sorts the column values alphabetically.</p> <p>other--Sorts the column values using a custom algorithm defined as a JPO and specified in the Sort Program setting for the column, or uses settings defined in Sort Type.</p>

Refreshing the Table After Adding, Editing, or Deleting Objects

You can use the `refreshTablePage()` JavaScript API method to refresh the table to update the content after adding a new item in the list, updating the selected item in a table, or deleting selected items in a table.

Call this function with reference to the top window object. For example, to refresh the table page, call:

```
// JavaScript Code  
top. refreshTablePage();  
// JavaScript Code
```

You can not use the `refreshTablePage()` method to update the `detailsDisplay` (the right side frame of `treeDisplay`) or the content frame.

Object or Relationship ID of a Selected Table Item

Once you have configured a table page to display objects and/or relationships in a table, you will typically want to add links to the page so users can work with the objects in the table.

You should display links that apply to objects or relationships in a table as hyperlinked data in the table or as toolbar items at the bottom of the page. Toolbar items should be commands that are independent of the table data, such as a Create New or Show Report link.

The JSP or JPO that gets called when one of these links is clicked needs the object or relationship ID(s) to work on. This table summarizes how the dynamic UI components make the IDs available to the requested page depending on the kind of link that is used.

If the table page link is:	Clicking the link results in:	For details, see:
Hyperlinked data within the table	<p>The objectId, relId or parentOID parameter automatically being put into the request. The availability of the parameter name depends on whether the column displays business object or relationship data.</p> <p>The objectId and relId are values for the selected row(s) and parentOID is the parent objectId passed to the Table page.</p>	Table Columns
A toolbar menu command that requires one or more selected table rows (selected using check boxes or a radio button in the left table column)	<p>The emxTableRowId parameter is available to the requested URL as long as the Submit setting for the toolbar item is True. The emxTableRowId parameter contains:</p> <ul style="list-style-type: none">For a table generated by a temp query Inquiry or JPO program returning object ID: one or more object IDs (one for each checked checkbox)For a table generated by an expand Inquiry or a JPO program returning object ID and rel ID: one or more object ID, relationship ID pairs (one for each checked check box or one for the selected radio button). The relationship ID and object ID values are separated by a pipe character.	For instructions on configuring the toolbar item, see Toolbars .

When programming custom JSPs, make sure you use built-in parameters instead of custom parameters. For more information, see [Parameters for Table Objects](#).

Table Columns

You need to configure all the columns that will be added to a table.

In this section:

- [Icons in Column Headings](#)
- [Settings for Table Column Objects](#)
- [Table Column Data Definition](#)
- [Editable Table Columns](#)
- [Sort Orders for Table Columns](#)
- [Group By Column](#)
- [Dynamic Columns](#)
- [Table Column Calculations](#)
- [Units of Measure for Columns](#)
- [Improving Performance of Table Columns](#)

Icons in Column Headings

This section describes how to define a column heading to include an image in addition to or instead of text.

The [Parameters for Table Objects](#) section includes the Heading parameter to define the column heading. Usually, the actual text of the column heading is placed in a string resources file to allow for internationalization, and the parameter passed to the JSP is the name of the string resource.

Although the table component does not require that parameters passed to the JSP be XHTML-compliant, the structure browser component does have this requirement. ENOVIA recommends that parameters for both tables and structure browsers be XHTML-compliant.

To be XHTML-compliant and work within the framework, the image tags provided as part of the column header name must:

- Use double-quotes for attribute values
- Include closing tags (can include the / close within the tag instead of)
- To display hover text, use the title attribute instead of alt
- To align the image, use the align attribute in a div tag that wraps the img tag, instead of using the align attribute within the img tag

For example,

```
<div align="center"></img></div>
```

This example shows a String Resource defined to use an image:

```
emxFramework.SBHeader.Type = 
```

This example shows a column heading that includes both text and an image:

```
emcFramework.SBHeader.OrderName =  Name
```

The column width is dynamically determined based on the header text. If there is no header defined, or the header is longer than 150 pixels, the column width defaults to 150 pixels.

Settings for Table Column Objects

This table lists and describes the settings for table objects.

Note that the name and values are case sensitive.

Setting	Description	Accepted Values/Examples
Access Expression	Controls access to table columns. For details, see User Access to UI Components .	--
Access Function		
Access Mask		
Access Program		
Additional Query	When Type Ahead is configured on the column (by setting the RangeHref to emxFullSearch.jsp or using a predefined Type Ahead Chooser), this setting defines additional field/selection critiera used to restrict the selection list. See About Automatic Type Ahead .	<FieldName1>=<select expression1>:<FieldName2>=<select expression2>:
Admin Type	Use to translate columns whose values are administrative types. Based on the administrative type, the value is translated and presented. For attributes, you must provide the symbolic name of the attribute, which starts with "attribute_" and is followed by the attribute name with no spaces.	Type State Role Relationship attribute_UnitOfMeasure
Allow Manual Edit	When true, users can manually edit the table column. Applicable only when the range parameter is set to a URL or when the setting format is assigned to date or for fields of type combobox. It is ignored in all other cases. When this setting is true, the Admin Type setting is ignored.	false (default)--Manual entry is not allowed. true--Manual entry is allowed.
Alternate OID expression	By default, when a column's data is configured to show as a hyperlink using the href parameter, the system passes the ID of the object for that row in the objectId parameter. Using this setting, you can configure the hyperlink so a different objectId is passed. The system passes the ID for the object returned from the expression defined in this setting.	\$<to[relationship_NewPartPartRevision].from.id> \$<to[relationship_EBOM].from.id> For more information on configuring a column using an alternate object ID's expression, see Column Values Using Alternate OID in href and Select Expression and Column Values Using Alternate OID and Type Icon in href with Select Expression .
Alternate Policy expression	This setting is required to display the state of any connected objects and show the translated value. This setting is applicable only when the Admin Type setting is set to State.	Any select expression that evaluates to a policy of a connected object.
Alternate Type expression	When the Show Alternate Icon setting is true, this expression obtains the object type. Based on the obtained type, the corresponding icon is displayed. For more information on configuring a column using an alternate type expression, see Column Values Using Alternate OID and Type Icon in href with Select Expression .	\$<to[relationship_NewPartPartRevision].from.type> \$<to[relationship_EBOM].from.type>
Auto Filter	Determines whether users can filter the table rows based on data in the column. If at least one column in the table has Auto Filter set to true, the Filter  tool displays in the page toolbar. You cannot apply this setting to any column that	true--The Filter tool displays in the page toolbar. When clicked, the column is available on the Auto Filter Selection page. false (default)--The column is not available in the Auto Filter Selection page.

	<p>has multiple values for a single data column cell.</p> <p>If the column contains an attribute configured with a dimension, the table is filtered based on the display value, not the normalized value.</p> <p>For more information on table filtering, see Methods for Filtering a Table.</p> <p>Configuring many columns for auto filtering or applying auto filter to inappropriate columns can adversely effect performance when viewing the filtered list. For optimum performance, define a limited number of columns with Auto Filter. Also choose columns that have a limited set of possible values on which the user may want to filter the table data. For example, state and owner columns are good candidates. Columns such as description and name are inappropriate because of the unlimited number of resulting values.</p>	
Calendar Function	Use to specify the name of the method in the JPO specified in the Calendar Program setting that retrieves the non-working days based on the calendar defined for the location.	The name of a function in the Calendar Program JPO, such as: <code>getNonWorkingDays</code>
Calendar Program	Use to specify the name of a JPO that contains a method to get the non-working days for a calendar.	The name of a JPO, such as: <code>emxWorkCalendar</code>
Column Icon	<p>Use to display an icon for the column's data instead of other data. Required when the setting "Column Type" is assigned to "icon".</p> <p>Also see Column Values as Icons.</p>	<p>Name of an image file such as:</p> <p><code>images/NewWindow.gif</code></p> <p><code>images/EditItem.gif</code></p>
Column Type	<p>Use the setting "Column Type" when no expression is defined for the column data.</p> <p>For recommendations on improving the performance of table columns whose data is generated with a JPO program (Column Type set to program or programHTMLOutput) and for a sample JPO program to get column data, see Improving Performance of Table Columns.</p>	<p>The column type can be one of the following:</p> <p>program--The values for this column are obtained from a program (JPO). With this setting, the program and function name are required as settings. For more information on configuring a column using a program, see Column Values as Program Output.</p> <p>programHTMLOutput--Same as the "program" setting above, except the column value output is in HTML format. Column values are placed in the table cell between <code><td></code> and <code></td></code> tags. This setting ignores other column settings such as Show Type Icon, href, format, and Alternate OID expression.</p> <p>icon--Used when the column values are shown as an icon. The setting "Column Icon" must be defined with the icon to be displayed. For more information on configuring a column using a program, see Column Values as Icons.</p> <p>image--Used when the column values are images. When used, the column shows the primary image associated with the business object (cannot be used with Structure Browser).</p> <p>checkbox--Used when the column values are check boxes shown enabled or disabled based on the access to the object in that row. This access can be based on business logic and defined in a JPO or it can be role-based and defined by assigning roles to the column. If the check boxes have no access restrictions, this setting is not required and you can create check boxes by passing in the parameter selection=multiple to emxTable.jsp. Also see Column Values as Check Boxes.</p> <p>separator--Used to define a column of white space between standard data columns. A separator is especially useful to separate two groups of columns.</p>

		file--Shows  which hyperlinks to a Quick File Access page listing files checked into or connected to the object. The optional Relationship Filter setting defines how to locate related files. If Common Components is not installed, this value is ignored.
Comparable	Used only for a table showing a structure compare report. Defines whether the column can be used as a comparison criteria in a structure compare report. By default, all columns are comparable except for Column Types of: Image Icon Separator	true (default) false
Compare Report	Used only for a table showing a structure compare report. Defines whether to show the column in the structure compare report. By default, all columns are shown except for Column Types of: • Image • Icon • Separator	hide show (default)
Currency Converter	Specifies that the currency shown in the column needs to be converted when the Currency Converter runs. When this setting is true for any column in a table, the table page automatically includes the Currency Converter tool. If this setting is not included, the value is assumed to be false and the target page does not display the Currency Converter tool.	true false (default)
Currency Expression	The name of the currency to convert from. Required for columns whose values are monetary and that can be converted from one currency to another using the Conversion tool and defined exchange rates. The value should be a select clause that returns the value for the Currency attribute for a specific object or relationship, or the actual value. The currency to be converted from should be the currency in which the user entered the data in (the As Entered currency).	For example, to convert currency data for an RFQ Quotation, the following select clause returns the supplier's currency format. to[Supplier Line Item].attribute[Currency]
Display Format	Specifies the number of the date format for any column where the value of the format setting is "date." See Date/Time Fields in Forms and Tables .	These are Java standard values of Date Format to display a date in a specific format. 3 - SHORT (12/12/52) 2 - MEDIUM (Dec 12, 1952) 1 - LONG (December 12, 1952) 0 - FULL (Tuesday, December 12 1952 AD) Default is set in emxSystem.properties: emxFramework.DateTime.DisplayFormat=MEDIUM. emxSystem.properties uses words, but the Display Format setting uses numbers.
Display Time	Controls whether the time is displayed along with the date for columns whose format is set to date. If no time zone preference is set, then the DateTime is shown in the browser's time zone. The time is shown in terms of GMT+/- hh:mm, (e.g., Saturday, August 21, 2004 12:45:00 PM GMT-04:00). To get the time in a format like EST or PDT, set the time zone preference to a specific zone. See Date/Time Fields in Forms and Tables .	true false Default is set in emxSystem.properties for the property emxFramework.DateTime.DisplayTime = false
Dynamic URL	When enabled, users can enter URLs or mxLink	enable (default)

	values and the values will display and function as hyperlinks.	disable
Edit Access Mask	<p>Use to control cell-level access in an editable table.</p> <p>The system performs the access check on all objects in one database call while retrieving the column values. If any cell does not have the specified access, then the cell is shown as read only.</p>	Any access mask, for example: modify, connect
Editable	<p>Use to indicate whether the column is displayed as editable or read only. Only applies for Edit mode. View mode ignores the setting.</p>	<p>true--Users can edit the column data when shown on the Edit mode.</p> <p>false (default)--Users cannot edit the column data when shown in the Edit mode. The column looks like it does in View mode except it is never hyperlinked.</p>
Effective Date Expression	<p>Use for columns whose values are monetary and that can be converted from one currency to another using defined exchange rates.</p> <p>Set the value to a select clause with a value for the Effectivity Date attribute on a specific object or relationship. The system uses this date to get the currency conversion whose rate period falls within this date. If this setting is not added, the current date is used.</p>	<p>For example, for currency data for an RFO Quotation, the following select clause returns the effectivity date.</p> <p>to[Supplier Line Item].attribute[Effectivity Date]</p>
Export	<p>Specifies whether column data is exported or not. Use this to change the export value on a column-by-column basis.</p> <p>If the column data is an attribute configured with a dimension, the exported data shows the value/units as displayed, not the normalized values.</p> <p>By default, all column types except programHTMLOutput are exported. If you want to additionally include programHTMLOutput, or exclude other column types from the export, change this setting.</p> <p>See Column Values as Program Output.</p>	<p>true false</p>
Field Type	Use only in edit mode while updating the table data.	<p>basic--The column displays basic information for the business object including: name, type, originated, policy, etc. Specifying basic as the field type is only needed when the column is editable. The only editable basic information is: type, name, revision, current, policy, description, owner, vault.</p> <p>attribute--The column displays values for an attribute on the business object, such as Originator or Weight.</p>
format	<p>Specifies the format to display the column data.</p> <p>If at least one column has the format set to currency or UOM, the Conversion tool displays in the table page's page toolbar. When a user clicks the tool, the system opens a new window and displays all column data defined with format=currency and UOM to the currency and unit of measure selected in preferences. For information on the currency and unit of measure preferences, see About Currency Conversions and About Unit of Measure Conversions.</p>	<p>alphanumeric--Uses the value stored in the database as is without units (for attributes configured with a Dimension).</p> <p>date--Displays the column values as a date. Uses the tag lib "emxUtil:IzDate" to format the display.</p> <p>currency--Displays the column values as currency.</p> <p>UOM--Displays the column values as Unit of Measure and enables the Unit of Measure conversion interface.</p> <p>email--Displays the column values as an email address. When a user clicks the email address, the email editor configured in the client is presented.</p> <p>numeric--Displays the column values as numbers. To perform calculations or graphically analyze the data on a column of string attributes that have numerical values, numeric must be the column type</p>

		user--Displays the user's name in the format "Lastname, Firstname".
function	The name of the method to call within the JPO program specified in the program setting. This method gets the column values if the setting "Column Type" is set to "program" or "programHTMLOutput" or "checkbox". See Column Values as Program Output .	getAssignedBuyerDesk getPackageAccess getParentPart getCurrentState
Group By	Groups rows in the table based on the value in this column. See Group By Column .	true
Group Header	Defines header text to display over several consecutive columns. For example, if you want a group header over three consecutive columns, add this setting to each column and assign the same value for each. You may want to separate grouped columns using a separator column, which is just a column of white space. Add a separator using Column Type=Separator. To see an example, see Grouped Columns and Column Separator .	Static text or string resource id.
Help Marker	Specifies the name of the help marker to call for context-sensitive help. In the href URL called when the column data is clicked, the system passes a parameter called HelpMarker and includes the marker text specified for this setting.	The naming convention for help markers "emxhelp" followed by the object or feature and then the action, for example, emxhelproutecreate and emxhelpprojectedit. The marker is all lowercase with no spaces.
Image Size	When the structure browser includes a Column Type of image, this setting defines which size of the primary image associated with the business object should display in the column and must use the symbolic name. The pixel dimensions for these sizes are defined using the emxComponents.Image.SizeType property (see the <i>Live Collaboration Administrator's Guide</i>) where Size is the Small, Medium, or Thumbnail.	format_mxSmallImage (default) format_mxLargeImage format_mxMediumImage format_mxThumbnailImage
Input Type	Used only for tables in Edit mode. Specifies the type of HTML control to display for user input. You can also designate the size of the input boxes to allow for appropriate spacing. This is important for short numeric entry fields.	textbox--This is the default. The column has a single-line box for typing text. textarea--The column has a multi-line box for typing text. combobox--The column has a drop-down list of options and users can only select one value. Use combo boxes for attributes that have defined ranges and for attributes whose ranges are determined with a Range Helper URL. To see an example of a field with a combo box, see Column Values Editable from Combo Box, Values from JPO .
Mass Update	Used only for tables in Edit mode. If set to false for the column, the column is not available to the Mass Update feature, but can be edited using the inline editing (one table cell at a time). If set to true, the column is used by the Mass Update feature.	true false
Nowrap	If the setting is true for any column, then the text in that column will never wrap. The default is false.	true false
Popup Modal	If the setting Target Location is set to popup, you can configure the window as modal or non-modal.	true--the popup window is modal false--the popup window is non-modal If the setting is not specified, the window is non-modal.
Printer Friendly	Passes the PrinterFriendly parameter and the entered value to the JSP specified in the column's href URL. JSPs that use the PrinterFriendly	true (default) false

	<p>parameter, such as emxTable.jsp and emxForm.jsp, show the Printer Friendly tool when the setting is True and hide it when False. If the setting is not included, emxTable.jsp and emxForm.jsp show the tool by default. Users can click the tool to get a version of the current page that can be printed with the browser's Print button.</p> <p>Note that you can also specify the PrinterFriendly parameter in the href URL for JSPs that use it.</p>	
program	<p>The name of the JPO program to get the column's data. This program gets the column values when the setting "Column Type" is set to "program" or "programHTMLOutput" or "checkbox".</p> <p>Using a JPO to populate column data is recommended only when you cannot use a select expression.</p> <p>For more information on configuring a column using a program, see Column Values as Program Output.</p>	emxSCSBuyerDesk emxTMCPackages emxENCParentPart emxCommonTableUtil
Range Function	<p>Use to specify the name of the method in the JPO specified in the Range Program setting. See the Range Program setting below for more information.</p>	The name of a function in the Range Program JPO, such as: getAssignedRange getClassificationRange getPartUOM
Range Program	<p>Use to specify the name of a JPO that contains a method to get the column value ranges (choices). A range program is used only when the Input Type setting is combobox or popup.</p> <p>The corresponding method which runs to get the values of the column should not have any HTML code.</p> <p>To see an example of a field configured with a range program, see Column Values Editable from Combo Box, Values from JPO.</p>	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
*Registered Suite	<p>The application the column belongs to. The system looks for files related to the column in the registered directory for that application, which is specified in emxSystem.properties.</p> <p>Based on the application name, the system passes the following parameters in the href URL:</p> <ul style="list-style-type: none"> suiteKey emxSuiteDirectory StringResourceFileId 	<p>The value cannot contain spaces, for example, EngineeringCentral or Framework. Set the value to the suite name as defined in the key eServiceSuites.DisplayedSuites in emxSystem.properties. If the suite name starts with "eServiceSuite" you can skip this prefix and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", you can assign the word "EngineeringCentral" as "Registered Suite".</p> <p>The system passes the "suiteKey" parameter in the href URL that is called when the user clicks the column data. The value for the parameter is the property name from emxSystem.properties that maps to the setting's value.</p> <p>If <code>Column Type=file</code>, this value must be Components.</p>
Relationship Filter	<p>Optional setting that defines a comma-separated list of relationship select expressions to get the file lists checked into or connected to the Type object.</p>	from [<relationship_ReferenceDocument>].to.id, to [<relationship_ReferenceDocument>].from.id,...,..
Required	<p>When creating or editing the value in the column, this setting defines if a value must be entered or is optional. If true, a value must be entered; if false, the value is optional.</p> <p>Some ENOVIA products might include properties that define whether an attribute is required for an</p>	true false

	<p>object. If so, the true/false value for the column for that attribute should match the true/false value for the property.</p>	
RMB Menu	<p>Specifies the name of the right-click menu to associate with the component.</p> <p>For a type-specific menu, such as type_Part, defines the type-specific right-click menu for that type.</p> <p>For a table column, specifies the right-click menu for that specific column.</p>	<admin menu name>
Show Alternate Icon	<p>If the column value display with the icon is different from the current row's object icon, set this setting to true. To get the right alternate icon, set the Alternate Type expression setting with the expression to obtain the object type.</p> <p>For more information, see Column Values Using Alternate OID and Type Icon in href with Select Expression.</p>	true false
Show Type Icon	<p>If true, the column value displays along with the object type icon as defined by the emxFramework.smallIcon property in emxSystem.properties. The icon displays to the left of the column data.</p> <p>If no property is defined for the type's icon, the system looks for a property defined for the parent type, then grandparent type and so on. If no property is defined for any type in the hierarchy, the system uses the default icon specified in the emxFramework.smallIcon.defaultType property.</p>	true false (default)
Sortable	Controls whether users can sort the table based on data in the table column. If a column is sortable, users can click the column heading to sort the table based on that column. If the column attribute is configured with a dimension, sorting is done based on the normalized value regardless of the displayed value.	true (default) false
Sort Direction	Specifies the sort order for fields whose type is combobox or listbox.	ascending (default)--Sort a to z or 0 to n. descending--Sort z to a or n to 0. none--The option list is not sorted.
Sort Program	<p>Specifies the JPO program name that contains a custom sorting algorithm. This setting is used only when the column parameter sorttype is assigned to "other".</p> <p>For guidelines and parameters or the custom sorting program and sample programs, see Guidelines for Custom Sort JPO.</p>	Name of a JPO added as a program in the database, such as: FindNumberSort CustomSort
Sort Range Values	<p>Enables or disables the sorting of range values in combobox or listbox controls. When enabled, the list is sorted based on the datatype and using the direction defined by the Sort Direction setting. When disabled, no sorting is done on the list.</p> <p>In a drop-down, range values populated based on an attribute expression will be sorted based on the attribute's datatype. Range values populated using the Range Program and Range Function settings will be sorted alphanumerically.</p> <p>If the Range Program or Range Function returns numeric or date values, the alphanumeric sort will not be appropriate. The program or function should sort the values in the required order, and this setting should be disabled.</p> <p>This setting does not apply to fields or columns</p>	enable (default) disable

	<p>configured for attributes associated with a dimension. Dimension ranges are always sorted as alphanumeric in ascending order.</p> <p>This setting cannot be used in custom JSPs that use the editOptionList taglib. For this specific situation, you can use the sortType attribute for that tag.</p>	
Sort Type	<p>Determines how to sort the column. You must include the MQL <code>sorttype</code> subclause.</p> <p>You can use <code>Sorttype</code> on its own to sort numeric columns (<code>sorttype = numeric</code>)</p> <p>OR</p> <p>You can use <code>sorttype</code> in conjunction with the Sort Type setting. Two examples:</p> <p>(<code>sorttype = other AND Sort Type = integer</code>)</p> <p>(<code>sorttype = other AND Sort Type = real</code>)</p>	date integer real
Target Location	<p>Controls where the page specified in the href parameter appears or is targeted.</p> <p>When using slidein as the Target Location and the Popup Modal setting is true, then the content window and the global toolbar are grayed out and disabled until the slidein window is closed.</p>	content--Page replaces the content frame. popup--Page appears in new window. The window modality can be set with the Popup Modal setting. _top--Page replaces the entire body of the browser window. listHidden--A hidden frame within the table frameset. Use this frame for Target Location for any processing in the context of a table. hiddenFrame--The frame called hiddenFrame is part of the top level Navigation window frameset and can be used to submit the Table frame and carry out background processing. This frame is not available for popup windows though so listHidden is a better frame to use. slidein--Page appears in a slidein frame (slides in along the right side of the window). You can set this value to any valid frame name available to the table body frame.
Tip Page	<p>Specifies whether the target page should include the Tip Page tool. In the href URL called when the column data is clicked, the system passes a parameter called TipPage and includes the URL specified for this setting. If this setting is not included, the target page does not display the Tip Page tool.</p>	Name of a custom html or JSP page, including any path. The starting point for the directory reference is the content directory. For example, if you want to call an html file in ematrix/doc/customcentral and the content directory is ematrix/customcentral, you would add this parameter to the table.jsp: TipPage=../doc/customcentral/tippage.html
Type Ahead Mapping	<p>When Type Ahead is configured on the column (by setting the RangeHref to emxFullSearch.jsp or using a predefined Type Ahead Chooser), this setting maps that field to the corresponding indexed field in config.xml. See About Automatic Type Ahead.</p>	NAME LASTNAME,FIRSTNAME,USERNAME
Type Ahead Validate	<p>When Type Ahead is configured on the column, this setting determines if the application should restrict the user to selecting one of the suggested values (if true), or if the user can enter any data in the field without Business Process Services validating that data (if false).</p>	true false (default)
Type Icon Function	<p>Specifies the method in the JPO specified in the Type Icon Program setting that retrieves an icon to show in addition to the Alternate Icon or Type Icon (enabled by Show Alternate Icon or Show Type Icon settings).</p> <p>If both the Show Alternate Icon and the Show Type Icon settings are set to false, any value for this setting is ignored.</p>	Method Name
Type Icon Program	<p>Defines the program that contains the function specified using the Type Icon Function setting.</p>	JPO Name

UOM Expression	<p>Specifies the name of the Unit of Measure to convert from. Required for columns whose values are measurements that can be converted from English units to Metric units or vice versa.</p> <p>The value can be a select clause that returns the value for the Unit of Measure attribute for a specific object or relationship or the actual value. The unit of measure to convert from should be the unit of measure that user entered (the As Entered unit of measure).</p>	<p>For example, for measurement data for RFQ Quotations, the following select clause returns the supplier's unit of measure format.</p> <p><code>to[Supplier Line Item].attribute[Unit of Measure]</code></p>
Update Function	Specifies a method name in the JPO given in the Update Program setting.	<p>The name of a function in the Update Program JPO, such as:</p> <p><code>setAssignedBuyerDesk</code> <code>setPackage Access</code> <code>setPartClassification</code></p>
Update Program	<p>Specifies a JPO that contains a method to set the column value when the column is displayed on an Edit mode table and when the user clicks Done. Use only when the Field Type setting is program or programHTMLOutput.</p> <p>Using an update program is recommended only when the Field Type is not attribute or basic.</p> <p>See Using JPO to Update Table Values for more information.</p>	<p>The name of a JPO, such as:</p> <p><code>SCSBuyerDeskForm</code> <code>TMCPackagesForm</code> <code>ENCPartForm</code> <code>AEFUtilForm</code></p>
Validate	Specifies the method to be invoked for validating any cell.	<p>The name of a method, such as:</p> <p><code>checkUniqueName</code></p>
Validate Type	Validates any column value with any specific characters defined in emxSystem.properties.	<p>Basic or Restricted--The column values are validated against the value of <code>emxFramework.Javascript.BadChars</code>.</p> <p>Name--The column values are validated against the value of <code>emxFramework.Javascript.NameBadChars</code>.</p>

*Required Setting

Table Column Data Definition

This section describes how to use the parameters and settings to define the data for columns. A table column can contain information about a business object or relationship, data from a program, check boxes, images, or icons.

In this section:

- [Column Values as Select Expressions](#)
- [Column Values as Program Output](#)
- [Column Values as Check Boxes](#)
- [Column Values as Images](#)
- [Column Values as Icons](#)
- [Column Values Using Alternate OID in href and Select Expression](#)
- [Column Values Using Alternate OID and Type Icon in href with Select Expression](#)
- [Column Values Including Additional Icons](#)
- [Column Values as Units of Measure](#)
- [Column Values as Quick File Access](#)
- [Grouped Columns and Column Separator](#)
- [Column Values as Special Information from Business Object](#)

Column Values as Select Expressions

You can get column data based on a select expression applied to a set of OIDs or RELIDs.

The column definition must include a valid select expression for a business object or relationship.

Some examples of select expressions include:

- type
- name
- \$<attribute[attribute_Originator].value>
- \$<attribute["Originator"].value>
- attribute["Originator"].value
- \$<attribute[attribute_FindNumber].value>

The following MQL script snippet defines a table whose column values are select expressions. The first two columns define expressions on business objects and the third has an expression on a relationship.

```
add table ENCParts system
column
    name Name
businessobject name
column
    name Originator
businessobject $<attribute[attribute_Originator].value>
column
    name FindNumber
relationship $<attribute[attribute_FindNumber].value>
```

Column Values as Program Output

You can obtain the column values based on specific business logic defined within a method in a Java Program Object (JPO).

Instead of a select expression, set the "Column Type" column setting to one of the following:

- Column Type = program: Use when the program results contain only the column value.
- Column Type = programHTMLOutput: Use when the program results contain the complete HTML tag (including the column value) to be displayed in the table column.

Both require a JPO for getting the column value results. You must configure the name of the JPO and the method for this column as settings for the table column:

- program = JPO_NAME: The JPO program object name to get the column values.
- function = JPO_METHOD_NAME: The JPO method name to return the results of column value.

When "columnType = program," the method that runs to get the values of the column should not have any HTML. If the method returns HTML, the same code is also written in the PrinterFriendly pages as links. For exports, the HTML tags are written to the exported file.

To return the HTML from the method, use the setting "Column Type = programHTMLOutput" instead of "Column Type = program." To get the column values exported when the Column Type is "programHTMLOutput," you must also set "Export=true" for the column.

To ensure that you do not return HTML or href links for PrinterFriendly pages and export files, the JPO method requires conditional logic to decide when to return the values with or without HTML. Use the parameter "reportFormat" (available to the JPO) for checking the condition.

For regular web page display, the reportFormat will be null or empty string. The reportFormat parameter will contain a valid value only for the export mode or Printer Friendly mode. Valid reportFormats are CSV, HTML, TXT and ExcelHTML.

The following sample code illustrates these methods:

```
public static Vector functionName (Context context, String args[]) throws Exception {
    Vector retVal = new Vector();

    String reportFormat = (String)paramMap.get("reportFormat");

    String name = "name to Displayed in the column";
    String outString = "html code";

    for (each object) {
        if(reportFormat != null && reportFormat.length() > 0)
        {
outString=name;
        } else {
outString=<a name=" + name + "href=../common/
emxTree.jsp?treeMenu=type_TechnicalSpecificationTemplate&mode=i
nsert&jsTreeID="+jsTreeID+"&objectId="+objectId+ "><img src=../
common/images/iconSmallSpecification.gif border=0>" + name + "</
a>";
        }

        // If anything special has to be done for specific report formats then users can
        // check reportFormat with corresponding string and write the code like the following if
        // needed

        if("CSV".equals(reportFormat)) {
            //special code for CSV format
        } else if("TXT".equals(reportFormat)) {
            //special code for TXT format
        } else if("HTML".equals(reportFormat)) {
            //special code for HTML format
        } else if("ExcelHTML".equals(reportFormat)) {
            //special code for ExcelHTML format
        }

        retVal.add(outString);
    }
    return retVal;
}
```

See [Improving Performance of Table Columns](#) for a sample program to get column values when Column Type is set to program and for recommendations on improving the performance of a programmatic table column.

The JPO for the columns must follow these rules:

- It must define a method with the exact name specified in the function column setting.
- The method must take an input argument as a HashMasp containing details of the object list to be processed. The data structure of this input parameter (HashMap) is defined below:

Key Name	Data Type	Description
----------	-----------	-------------

objectList	MapList	<p>This MapList contains a list of HashMaps with the values of OIDs and RelIDs. Every HashMap contains the value for OID and RelID belonging to one table row. The size of the MapList passed in equals the list of rows to be presented in the table.</p> <p>The structure of each HashMap belonging to a table row is defined below: ?</p> <p>Key "id" is assigned to the business object Id (OID) to be used in the current table row.</p> <p>Key "id[connection]" is assigned to the relationship id (RelID) to be used for the current row.</p>
paramList	HashMap	<p>This HashMap contains the all the request parameter key / value pair available in the JSP page level.</p> <p>An additional parameter "languageStr" is available to the HashMap. This key is assigned to the value with current browser language setting (like en, fr, ja etc). This value may be used for the internationalization purpose within the JPO program.</p>

- The method must return a vector of String or StringList, containing the column values to display. The vector size must be equal to the size of the input MapList.

Here is a template for the JPO function.

```

public static Vector methodName(Context context, String[] args)
throws Exception
{
    HashMap programMap = (HashMap) JPO.unpackArgs(args);
    MapList relBusObjPageList =
(MapList)programMap.get("objectList");
    HashMap paramMap = (HashMap)programMap.get("paramList");
    Vector columnValues = new Vector(relBusObjPageList.size());
    String bArr [] = new String[relBusObjPageList.size()];
    String rArr[] = new String[relBusObjPageList.size()];
    StringList bSel = new StringList();
    StringList rSel = new StringList();
    // Get the required parameter values from "paramMap" - if
    required
    String languageStr = (String)paramMap.get("languageStr");
    // Get the object elements - OIDs and RELIDs - if required
    for (int i = 0; i < relBusObjPageList.size(); i++)
    {
        // Get Business object Id
        bArr [i] =
(String)((HashMap)relBusObjPageList.get(i)).get("id");
        // Get Relationship Id
        rArr [i] =
(String)((HashMap)relBusObjPageList.get(i)).get("id[connection]");
    }
    // Add any Business object selects if required
    bSel.add("current");
    bSel.add("policy");
    bSel.add("attribute[Originator].value");
    // Add any Relationship selects if required
    rSel.add("attribute[Find Number].value");
    // Process the OIDs to get the results - if required
    BusinessObjectWithSelectList bwsl =
context.getSelectBusinessObjectData(bArr, bSel);
    // Process the RelIDs to get the results - if required
    RelationshipWithSelectList rwsl =
context.getSelectRelationshipData(rArr, rSel);
    // Code for processing the result data obtained - if required
    //Build the Vector "columnValues" with the list of values to
    be displayed in the column
    return columnValues;
}

```

Column Values as Check Boxes

There are a couple of ways to configure a column of check boxes.

- Pass in the parameter selection=multiple to emxTable.jsp to add a small column of check boxes on the left side of the table page. There is no access control for these check boxes so they are enabled for all users. For more information on the selection parameter, see [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#).

This parameter takes precedence over any check box column added to the table administrative object (using Column Type=checkbox). So you pass selection=multiple to emxTable.jsp, any check box column added to the table is not used. To use a check box column added to the table object, pass selection=none to the emxTable.jsp.

- Add a column with the setting Column Type=checkbox. For this type of column, the column value is a check box that is disabled or enabled based on specific business logic or based on the roles assigned access to the column.

If access is business logic based, and not role-based, the business logic to decide the access for every row is implemented in a JPO. The program and function settings must also be defined for this option.

The JPO is written in the same way as described in [Column Values as Program Output](#). The only difference is that the method must return Vector containing the strings either true or false. If the value of Vector elements contains true, the column cell displays with an active check box and if false, the column cell displays with a grayed check box.

Column Values as Images

The column value can be an image. The table column displays the primary image associated with the business object.

When an image is checked in, the Image Manager checks in the primary image, then generates and checks in multiple sizes of that image. You can define which size to use in the table column using the Image Size setting which takes a value of the symbolic name for Small (default), Thumbnail, Medium, or Large.

To configure a column to display images, use these settings:

- Column type: image
- Image Size: format_mxSmallImage, format_mxMediumImage, format_mxThumbnailImage, format_mxLargeImage

If you configure the column as an href value, clicking the image opens the image in a new window. If you do not configure an href value, clicking the image opens the Image Viewer in a new window.

See the *Live Collaboration Administrator's Guide* for details on the system properties that define defaults for images in configurable components. See the *Common Components User's Guide* for instructions on using the Image Viewer and Image Manager.

Column Values as Icons

The column value can be an icon.

The right-most column in most tables, which contains the popup window  icon and displays details for the current object, is an example of this kind of column. To configure a column so its data is an icon, configure the table column with Column Type=icon. Specify the icon to display in the column setting "Column Icon".

Column Values Using Alternate OID in href and Select Expression

Another way to configure hyperlinked column data is to use a select expression and an alternate OID (one different from the OID for the row's object).

You pass this alternate OID as parameter objectId as part of the href when the user clicks the column data. You can use the setting Alternate OID expression and assign it a valid select expression. The expression must be a select clause for obtaining an OID connected/related to the current object in the table row.

The Workspace Folder(s) and Package columns depict this kind of column data. The href for the hyperlinked workspace name, for example **DMD Pckg 1**, is appended with the objectId parameter. The value for the parameter is obtained from the select expression assigned to the Alternate OID expression setting. If the Alternate OID expression setting is not used in a column, the current row's objectId is appended to the href value. For example, the objectId parameter used in the Name column is the current row's objectId.



The screenshot shows a software application window titled "RFQs". The main area is a grid table with the following columns: Name, Round, State, Owner, Quote Requested By, Response Status, Package, and Workspace Folder(s). There are three rows of data:

	Name	Round	State	Owner	Quote Requested By	Response Status	Package	Workspace Folder(s)
	DMD 115	1	Started	Test Everything	Mar 11, 2011 9:00:00 PM	0/0	DMD Pckg 1	 
	DMD 526	1	Started	Test Everything	Mar 11, 2011 9:00:00 PM	0/0	DMD Pckg 1	 
	DMD 724	1	Started	Test Everything	Mar 25, 2011 9:00:00 PM	0/0	DMD Pckg 1	 

Column Values Using Alternate OID and Type Icon in href with Select Expression

In addition to using a select expression to show the hyperlinked name of an alternate objectID (as described in the previous section), the column can include the icon for the alternate object's type.

To include this icon, assign the Alternate Type expression setting to a valid select expression and assign the Show Alternate Icon setting to true. The select expression must be a select clause for obtaining a TYPE of the connected/related object. The system uses the type name returned by this select expression to obtain the icon name to display.

The ECOs column shows an example of column data that includes a type icon. The icon appears to the left of the object name. The icon was obtained based on the Alternate Type expression and Show Alternate Icon settings. This column is also configured to use the Alternate OID expression setting for the hyperlinked data.

	Type	Name	Rev	State	ECOs	EBOM	Weight	Originated	Vault	
<input type="checkbox"/>	Part	CM-200000-01	3	Release	ECO-007020		0.0	Apr 19, 2002 11:10:16 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200000-01	2	Release	ECO-007010		0.0	Apr 19, 2002 11:10:14 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200000-01	1	Release	ECO-007000		0.0	Apr 19, 2002 11:10:04 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200001-01	1	Release			0.0	Apr 19, 2002 11:10:06 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200002-01	1	Release			0.0	Apr 19, 2002 11:10:06 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200003-01	1	Preliminary			0.0	Apr 19, 2002 11:10:06 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200004-01	1	Preliminary			0.0	Apr 19, 2002 11:10:06 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200005-01	1	Preliminary			0.0	Apr 19, 2002 11:10:06 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200006-01	1	Preliminary			0.0	Apr 19, 2002 11:10:06 AM	eService Sample	
<input type="checkbox"/>	Part	CM-200007-01	1	Preliminary			0.0	Apr 19, 2002 11:10:06 AM	eService Sample	

This is the MQL code snippet for this use case:

```
//*****
column
    name ECOs
    label emxEngineeringCentral.Common.ECO
    user "all"
    businessobject
    "$<to[relationship_NewPartPartRevision].from.name>" 
        setting "Alternate OID expression"
    "$<to[relationship_NewPartPartRevision].from.id>" 
        setting "Alternate Type expression"
    "$<to[relationship_NewPartPartRevision].from.type>" 
        setting "Show Alternate Icon" true
        href "emxTree.jsp"
        setting "Registered Suite" "EngineeringCentral"
//*****
```

Column Values Including Additional Icons

A column can include an icon in addition to the object name using any of these methods described in this section.

- Icon based on the object type (set using the emxFramework.smallIcon.defaultType property defined in the *Live Collaboration Administrator's Guide*)
- Alternate icon based on a select expression (see [Column Values Using Alternate OID and Type Icon in href with Select Expression](#))
- JPO function to add an additional icon to the column or field based on the object's lifecycle state, or any other business requirement

The emxFramework.UIComponents.OverrideTypeIcon.Program=<JPO Name>:<Method Name> property in emxSystem.properties defines a global JPO:method used for all fields and columns that define the **Show Alternate Icon** or the **Show Type Icon** setting, and for all navigation trees. To use a different JPO:method for a specific column, use the **Type Icon Function** and **Type Icon Program** settings.

The JPO:method specified in the emxSystem.properties file and the one specified using the **Type Icon Function** and **Type Icon Program** settings determine the icon to show in addition to the type or alternate icon already defined. For this JPO to be called, either the **Show Alternate Icon** or the **Show Type Icon** setting must be set to true. If both of those settings are set to false, the default JPO and any **Type Icon Function** and **Type Icon Program** values are ignored.

You cannot use the additional icon if the value for the Column Type setting is any of these values:

- checkbox
- file
- icon
- image
- programHTMLOutput
- separator

The function that retrieves the list of custom icons to display in the column must return an image name or path for each id that is passed to it. The path, including the image name, must be specified relative to the ematrix\common directory. The image must be a gif image, and cannot include any spaces or special characters.

If the function returns:

- Image name--the image is used (you must provide all images not expressly installed with the ENOVIA products)
- Blank image name--the default image defined for the object type is used
- Invalid image name--the standard red X icon is used

The method signature is:

```
public static MapList getCustomIcons(Context context, String[] args) throws FrameworkException
```

The input args[] array contains a packed map with the entries defined in this table.

#	Key	Content
1	paramMap	A map containing key/value pairs of the request parameters originally passed to emxTable.jsp or emxIndentedTable.jsp.
2	columnMap or fieldMap	Information about the column or field
3	ObjectList	A MapList (derived from the ArrayList) of maps with information about each object, such as object id or connection id, in the table or structure browser.

If the column has also been defined with the Show Alternate Icon and Alternate OID expression settings, it is possible that the expression could return multiple values. In this case, the Type Icon Function must be passed the same list of resolved object ids, as shown in this example:

Key	Value Description
id	Object id of the table row
id[connection]	Relationship id of the table row
Alternate OID	List of Alternate Object Ids: 123.495.7238.7888 123.495.7238.7889

Key	Value Description
id	Object id of the table row
id[connection]	Relationship id of the table row
Alternate OID	List of Alternate Object Ids: 123.495.7238.7888 123.495.7238.7889

When the `Show Type Icon` setting is set to true, the MapList output (that specifies the name of the additional icon) from a function that processes alternate OIDs should look like this:

Key	Value
121.783.378.7774	completeState.gif
121.783.378.7734	startState.gif
Key	Value
121.783.378.7778	completeState.gif
121.783.378.7755	startState.gif

When the `Show Type Icon` setting is set to true for a specific column, the MapList output (that specifies the name of the additional icon) from a function that processes alternate OIDs should look like this:

Key	Value (Alternate HashMap)	
121.783.378.7774	Alternate OID	Value
	121.783.378.7722	completeState.gif
	121.783.378.7755	startState.gif
121.783.378.7734	Alternate OID	Value
	121.783.378.7711	completeState.gif
	121.783.378.7799	startState.gif

Sample Code for Type Icon Program/Function Used with Show Type Icon

```
Public static MapList getCustomIcons (Context context, String[] args) throws FrameworkException
{
    // unpack args array to get input map
    Map programMap = (Map) JPO.unpackArgs(args);
    // get object list
    MapList ObjectList= (MapList) programMap.get("objectList");
    MapList iconList = new MapList();
    String bArr [] = new String[ObjectList.size()];
    StringList bSel = new StringList();
    bSel.add(DomainConstants.SELECT_CURRENT);
    bSel.add(DomainConstants.SELECT_TYPE);

    // Get the object elements - OIDs and RELIDs - if required
    for (int i = 0; i < ObjectList.size(); i++)
    {
        // Get Business object Id
        bArr [i] =(String)((HashMap) ObjectList.get(i)).get("id");
    }
    // Get the required information for the objects.
    BusinessObjectWithSelectList bwsl
    =BusinessObject.getSelectBusinessObjectData (context, bArr,
    bSel);
    for (int i = 0; i < ObjectList.size(); i++)
    {
        String currentObjectid =(String)((HashMap)
        ObjectList.get(i)).get("id");
        // get the current state value
    }
}
```

```

String state = bwsl.getElement(i).getSelectData("state");
String type = bwsl.getElement(i).getSelectData("type");
HashMap retMap = new HashMap();
// Based on the object state add the required icon
//Pass the required information to this method and get the
required icon //name.
    String objectIcon = getIcons(context,typeName,state,
currentObjectid);
retMap.put(currentObjectId, objectIcon);
// Size of the iconList should be as same as ObjectList.
iconList.add(retMap);
}
// return final list
return iconList;
}

```

Sample Code for Type Icon Program/Function Used with Show Alternate Icon

```

Public static MapList geAlternateCustomIcons (Context context,
String[] args) throws FrameworkException
{
// unpack the incoming arguments into a HashMap called
'programMap'
Map programMap = (Map) JPO.unpackArgs(args);
    // Get the objectList
MapList relBusObjPageList= (MapList)
programMap.get("objectList");
    MapList iconList = new MapList();
    StringList bSel = new StringList();
    String state = new String();
    String typeName = new String();
// Add the Domain Constants as required.
bSel.add(DomainConstants.SELECT_CURRENT);
    bSel.add(DomainConstants.SELECT_TYPE);
for(int j=0;j<relBusObjPageList.size();j++){
    //Create a hashmap to contain each object id as key and
icon as value
    HashMap retMap = new HashMap();
    Map objectMap = (Map) relBusObjPageList.get(j);
        // Get the current Object id from the objectMap
String ObjectId = (String)objectMap.get("id");
// Define a Map which contains the alternateObject id  as key
and icon name as value
HashMap alternateMap = new HashMap();
// Get the alternate object id list from the objectMap
    StringList alternateObjectList =
(StringList)objectMap.get("Alternate OID");
//Validating if the objectlist have the alternate object ids.
/*If true then get the alternate object ids from the list and
get the corresponding icons and put these values in a alternate
map with alternate object id as key and icon as value. After
processing all the alternate object ids put this HashMap in a
returnMap where current object Id is the key and this map as
value so that the iconList size become same as ObjectList size*/
    if(alternateObjectList!=null){
String alternateObjects[] = new
String[alternateObjectList.size()];
    for (int k = 0; k < alternateObjectList.size(); k++)
        alternateObjects[k] = (String) alternateObjectList.get(k);
// Get the required informations for the alternate objects
BusinessObjectWithSelectList alternatebwsl =
BusinessObject.getSelectBusinessObjectData(context,alternateOb
jects, bSel);
    for (int k = 0; k < alternateObjectList.size(); k++) {
        if (alternatebwsl.getElement(k) != null) {
state = alternatebwsl.getElement(k). getSelectData
(DomainConstants.SELECT_CURRENT);
typeName = alternatebwsl.getElement(k). getSelectData
(DomainConstants.SELECT_TYPE);
        }
        String alternateOID = (String)
alternateObjectList.get(k);
//Pass the required information to this method and get the

```

```
required icon //name.
    String alternateIcon =
getIcons(context,typeName,state,altOID);
// alternateMap should contain altOID as key and icon name as
value.
alternateMap.put(alternateOID,alternateIcon);
} //end for
//Now put the original object id as key and the alternateMap as
a value in the //return Map.
retMap.put(ObjectId,alternateMap);
}
// iconList size should be as same as ObjectList size.
iconList.add(relMap);
}
return iconList;
}

public static String getIcons(Context context, String typeName,
String state,String objectAltId) {
    if (state.equalsIgnoreCase("state_name")) // eg
Preliminary
        return "image_name.gif";
    else if (state.equalsIgnoreCase("state_name")) // eg
Review
        return "image_name.gif";
    else {
        String typeIcon =
UINavigatorUtil.getTypeIconProperty(context,typeName);
        return typeIcon;
    }
}
```

Column Values as Units of Measure

For attributes configured with a dimension, the units of measure can be displayed in any of the units defined for the dimension.

See the *Business Modeler Guide* for instructions on defining dimensions. In addition, units of measure can be entered using any supported units as shown in the pull-down list.

Refer to [Selectable](#), for the options available for displaying units of measure information. For example, to include a column for a Weight attribute that includes kg, lb, and gm units, this selectable can be defined:

```
attribute[attribute_Weight].value
```

The returned value includes both the numeric value and the units, such as "10 kg".

If the user has set a preference (see [Configurable Preference Pages](#), then the column shows both the as-entered format followed by the preferred format in parentheses. For example, if the value was entered as "1.1 yd" and the user's preference is Metric, then the value in the column shows as:

```
1.1 yd (1 m)
```

For the preferred values to display, the units in the dimension must be mapped to English or Metric using the system modifier as described in the MQL Guide. If the mapping is not configured, no preferred values will display even if the units have been defined.

If you use a subselect expression to populate the column, such as `attribute[Weight].inputvalue`, then the column shows the result of the select expression and does not use the preference setting. To continue the above example, the column value would be:

```
1.1 yd
```

because that was the input value/units.

You do not need to update existing JPOs to use the Unit of Measure selectables. For example, a JPO was written using a selectable for an attribute not configured with a dimension. The JPO would use `attribute.value` as the selectable to retrieve the value, and that value would be what the user entered without any conversions. If the attribute was subsequently configured with a dimension, the framework intercepts `attribute.value` and replaces it with `attribute.inputvalue.toString`. The value returned to the JPO is the same as it would have been prior to adding the dimension to the attribute.

To display the normalized value without units, that is, the value actually stored in the database, you can use the `format=alphanumeric` setting (see [Settings for Table Column Objects](#)). To continue the above example, if the dimension is normalized on cm, then using this format setting would result in the column value of:

```
100
```

For columns displayed in editable tables, only the as-entered units of measure are shown.

Column Values as Quick File Access

You can configure a column to link to the Quick File Access configurable table page that lists the files checked into or connected to the object.

The optional Relationship Filter setting defines how to locate which files can be accessed from this column. To configure this column, use the `Column Type=file` setting. See the *Common Components User's Guide* for details of the Quick File Access page.

Grouped Columns and Column Separator

You can create a group of columns by adding a header over several consecutive columns and by separating the columns with a separator.

To specify a group header, add the setting Group Header for each column in the group. To add a separator, use the Column Type=separator setting. Here is an example of grouped columns with separators.

The screenshot shows a software interface titled "Defect Assessment Dashboard". At the top, there is a toolbar with various icons. Below the toolbar is a search bar containing the text "All". The main area is a grid-based table with several columns. A vertical blue line labeled "Group Header" spans multiple columns, indicating a group header. Another vertical blue line labeled "Separator" is positioned between two groups of columns. The columns are grouped into sections: "Basics", "Status", and "Reporting Information". The "Basics" section contains columns for Name, Title, Description, State, Priority, Severity, Owner, Originated Date, Originator, Reason For Rejection, Reported Against, and Reporting Organization. The "Status" section contains a single column for Reason For Rejection. The "Reporting Information" section contains columns for Reported Against and Reporting Organization. The first row of the table displays the column headers, and a single data row is visible below it.

Name	Title	Description	State	Priority	Severity	Owner	Originated Date	Originator	Reason For Rejection	Reported Against	Reporting Organization
DFT-0000001	test	test	Rejected	P1	G1	Test Everything	Feb 11, 2011	Test Everything	N/A	CHIP 1.1	Company Name

Column Values as Special Information from Business Object

You can configure a table column to show any information available to the method that obtained the object list.

The list generated by inquiry or JPO might contain more information than the OID and RelID. This special information may not be available at a later point for displaying as column values. For example, when doing an expand on a part for relationship EBOM, the level information is available, but cannot easily be obtained from the resulting object list.

To make this special information available for column values, you can obtain it when fetching the object list. You can store it in the object list map with a pre-defined key. To display these values in a column, configure a column as Column Type=program, use the Object List Map to obtain the values and display them as column values.

Editable Table Columns

This section shows examples of how to configure table columns to be editable.

The following topics are discussed:

- [Column Values Editable in Text Box](#)
- [Column Values Editable from Combo Box, Values from JPO](#)
- [Column Values as Editable Dates](#)
- [Using JPO to Update Table Values](#)

Column Values Editable in Text Box

You can let users edit column values by typing new or changed information in a text box. If the attribute specified for the field is configured with a dimension, a drop-down list shows next to the text box allowing the user to select any defined unit of measure. The current value of the field shows using the As Entered units of measure. Regardless of the units that the user selects, the value will be stored in the database using the default (normalized) units.

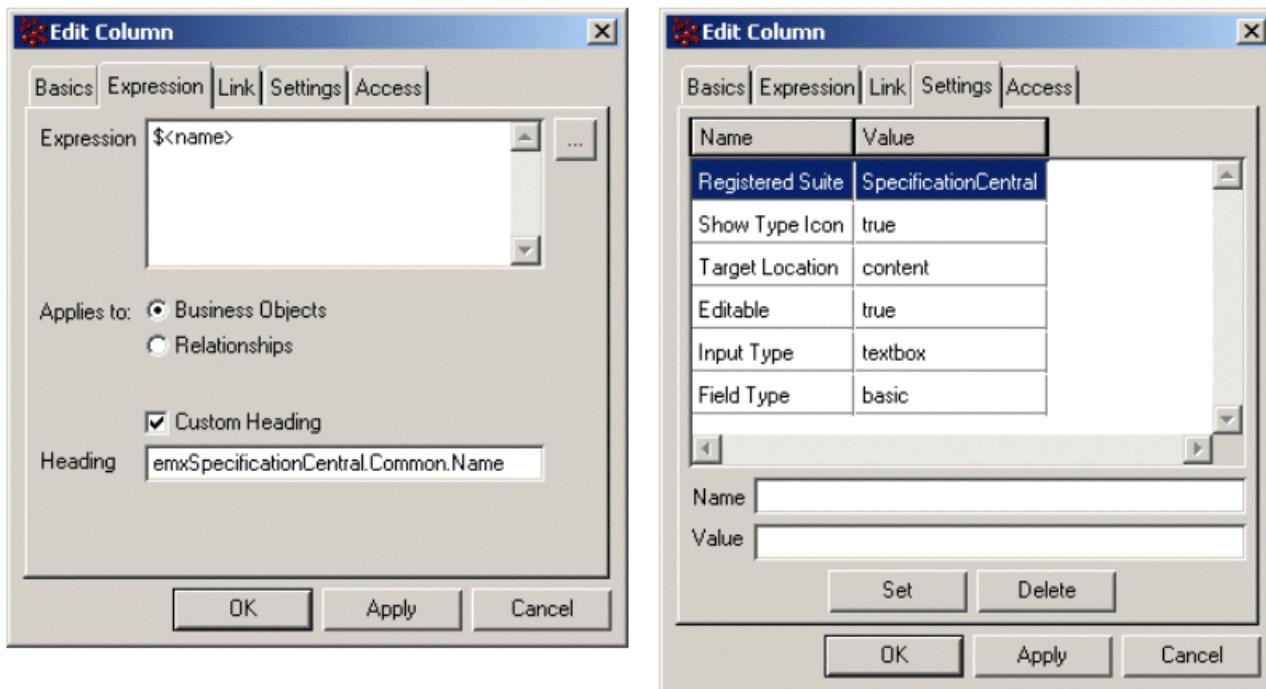
For example, to get data for a Description column, you can use a select expression applied to the current business object and let users edit the name using a simple text box. This graphic shows examples of a Description column for a table displayed in Edit mode.

Description	Invoice Number
Descr for Invoice AB-008	

To configure a column so the data is obtained from a select expression and displayed in an editable text box, use these parameters and settings

- Expression parameter--Enter the select expression to apply to a business object or relationship to get the column values.
- Applies to--Choose what the expression should be applied to: Business Object or Relationship.
- Field Type setting--When the expression is a business object basic or an attribute and the column is editable, include the Field Type setting. This enables the system to correctly update the basic or attribute property.
- Editable setting--Set to true to allow users to edit the column data.

These graphics show how you can configure the Name column in Business Modeler.



Column Values Editable from Combo Box, Values from JPO

When the set of possible values for the column is fairly small, you can represent them as combo box where users select the values. The following examples shows three columns displayed as combo boxes.

Business Unit	Region	Specification Office
BU 1	Region 1	Spec Office 1

BU 1 Region 1 Spec Office 1

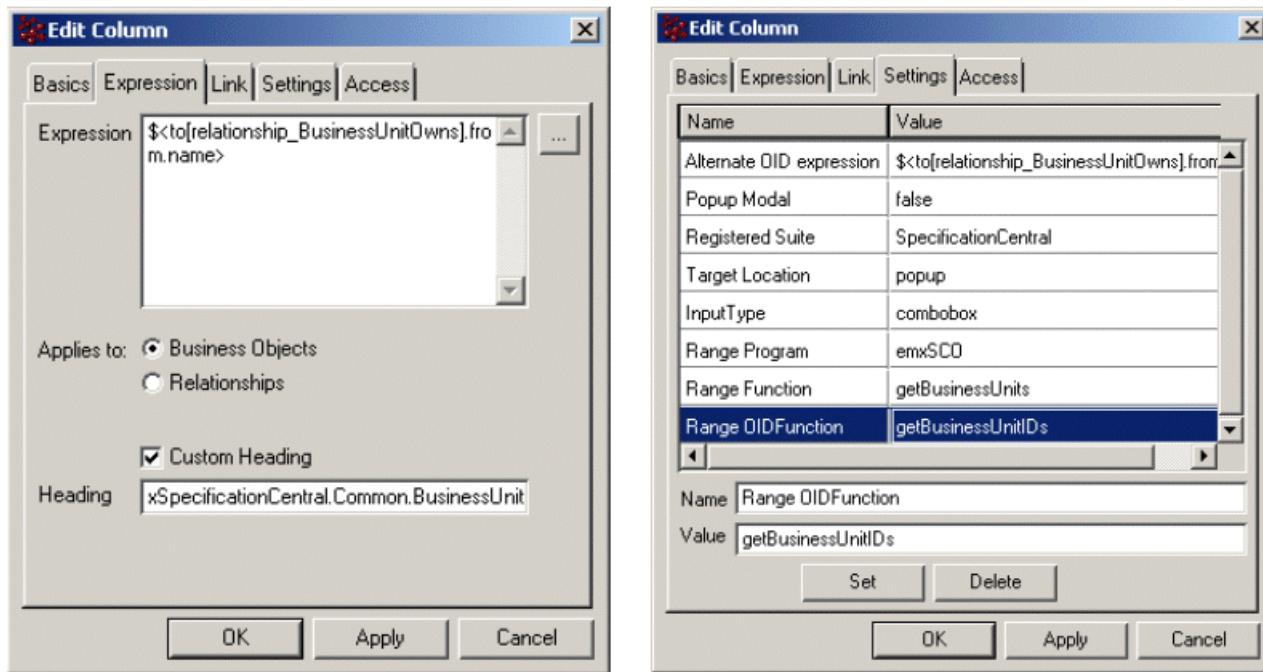
BU 1	Region 1	Spec Office 1
------	----------	---------------

You can populate combo boxes through a JPO method that obtains the values based on business logic.

To get column range values using a method in a JPO, use these parameters and settings

- Column Type setting--Set to `program` when program results contain only column values. Set to `programHTMLOutput` when program results contain the complete HTML tag (including the column value) to display in the column.
- Input Type setting--combobox
- Range Program setting--The JPO that obtains the choices.
- Range Function--The method that returns the choices in the object type string list.
- Editable setting--Set to true to let users edit the column data.

The following show a column configured to display editable choices in a combo box based on data retrieved via a JPO. The currently-selected value for the field displayed in the view-only version of the table was retrieved via an expression.



Column Values as Editable Dates

You can configure column values to display dates. You can let users change the date using a calendar chooser. The value must be a valid date string. The date displays based on the system and browser locale setting using the IzDate taglib. For example:

Planned Build Date (MM d, yyyy)	Actual Build Date
Apr 8, 2004	Apr 14, 2004
Apr 7, 2004	Apr 8, 2004

Ensure that the format of the value displayed in the column matches the format listed in parenthesis following the date control.

- The first column, Planned Build Date, uses these settings:

format = date

Editable = true

Allow Manual Edit = true

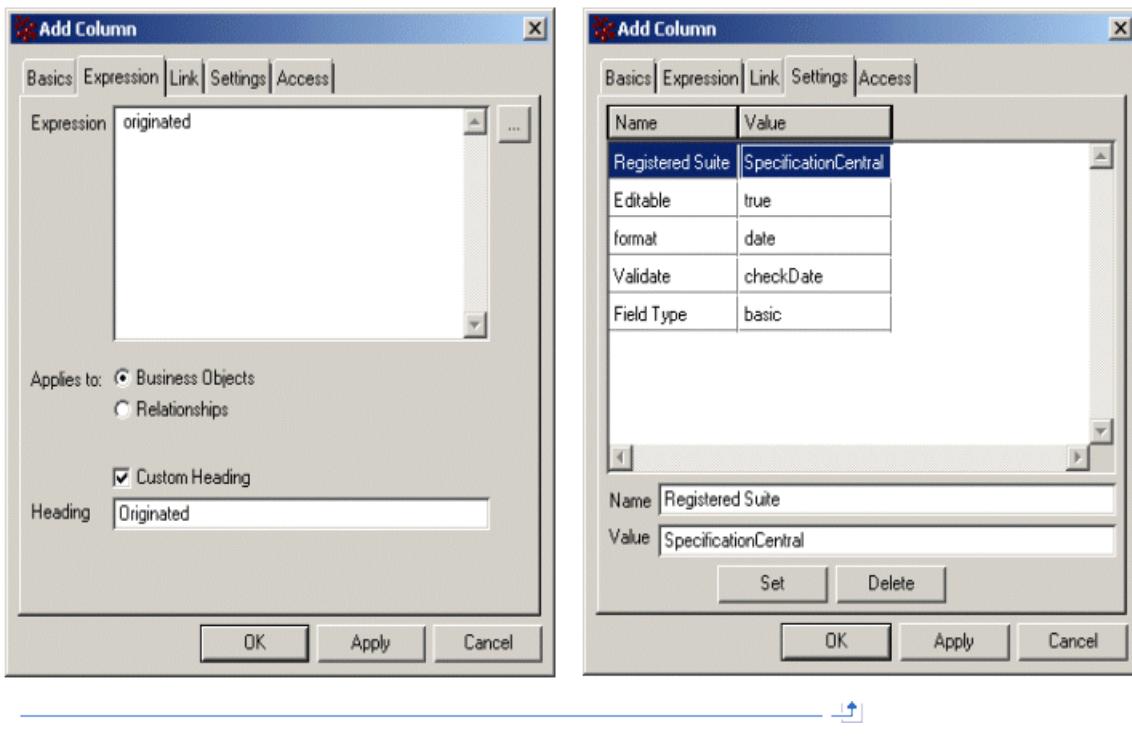
(The column header displays the date format depending on the browser locale setting.)

- The second column, Actual Build Date, uses these settings:

format = date

Editable = true

The following illustrates a column set up to display the originated date. It is set to validate the data.

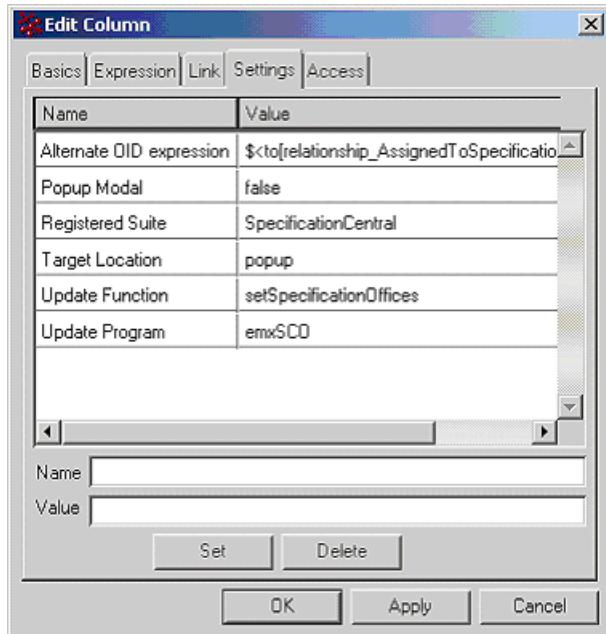


Using JPO to Update Table Values

Use JPOs to store the values of the related objects. See [JPO Guidelines for Updating Column Values](#).

Set these column settings:

- Update Program--The JPO containing the method that saves the values.
- Update Function--The name of the method that saves the data.



Sort Orders for Table Columns

You can define how table columns should be sorted when the user selects the sort tool.

Users can sort the columns by clicking on the header text link or the  toolbar button.  opens a dialog box allowing the user to choose up to 3 columns for sorting. Only columns defined with the Sortable=true setting can be selected. See the *Application Exchange Framework User's Guide* for details.

When the page displays the first time, the initial sorting is done based on the URL parameters `sortColumnName` and `sortDirection`. If you do not pass these URL parameters, the rows display in the order they are retrieved from the database. You can pass up to 3 column names with the `sortColumnName` parameter and 3 directions with the `sortDirection` parameter.

When the user opens the Sort by dialog box (by clicking  in the page toolbar), the 1st, 2nd, and 3rd column name fields are populated based on the values passed to the `sortColumnName` and `sortDirection` URL parameters. If less than 3 values are passed, only the values passed show populated in the dialog box and the remaining fields are blank.

If the user creates a custom table based on the system table for the table page, the sorting options selected by the user determine the default values for the Sort by dialog box. If the user clicks on one of the other column headers to sort, the system updates the page level `sortColumnName` setting to the new column.

For pages that include columns with the type `program` or `programHTMLOutput` with complex logic, the sort process may take a while. You can set the Sortable setting on the column to false if needed.

To define your own sorting mechanism, see [Sorting Programs for Tables](#).

Group By Column

You can use the Group By setting on a column to use the values in that column as headings (including an icon if the Show Type Icon = true setting is defined) under which all rows with a specific value in that column will be grouped together.

Any sorting done on the page will be done within the grouped rows only. Only one column in a table can have this setting, and the only valid value is `true`.

For example, this page shows a Type column defined with Group By = true.

Table Group By feature (Click on the name link of any Customer object)			
Categories	Edit	View	?
Name	Revision	Description	
AEFQE_SBCorpCustomer			
<input type="checkbox"/> Acme Widget	1	A Widget a Day Keeps the IRS Away	
<input type="checkbox"/> Amalgamated Broom	1	Brooms Unite!	
<input type="checkbox"/> Zirconium Keg	1	zasdfas	
<input type="checkbox"/> ZZ Tops & Novelties	1	By design - no Objects are Connected to this Customer	
AEFQE_SBItemSoftware			
<input type="checkbox"/> Ad-Aware SE	1	DataMiner Malware Annoyware Worm Adware Spyware	
<input type="checkbox"/> Beyond Compare	1	Select Left Side to Compare	
<input type="checkbox"/> FreeCell	1	That move is not allowed.	
<input type="checkbox"/> Snood	1	Great game, but annoying web site: www.snood.com	
<input type="checkbox"/> UltraEdit-32	1	Please enter registration key.	
AEFQE_SBItemDVD			
<input type="checkbox"/> Dr. Strangelove	1	Gentlemen, you can't fight in here! This is the War Room!	

You can define this setting for any type of column except file, image, separator, checkbox and icon. The Group By setting works in both View and Edit mode.

Dynamic Columns

You can dynamically add columns to a table depending on the existing data.

The following topics are discussed:

- [Dynamic Column Definition](#)
- [Guidelines for Adding Dynamic Columns to a Table](#)
- [Sample JPO Code](#)

Dynamic Column Definition

If you wanted to display a table that lists Buyers and all the Supplier Companies they work with, you could define the Buyer column, and then define the Supplier Companies columns to be added dynamically. The content of the columns would be a check mark or x or other indicator.

You can include multiple sets of dynamic columns in a table.

To dynamically add columns to a table, you need to write a custom JPO:method that retrieves the required columns and data for those columns. You use these settings on the column:

- Column Type = Dynamic
- Dynamic Column Program = JPO name
- Dynamic Column Function = JPO method name

The table can include any number of specifically added columns in addition to the dynamic columns. If the table only includes dynamic columns and an empty columnMap is returned, an error is presented to the user.

This is the required method signature:

```
public static List getCompanyColumns(Context context, String[] args) throws FrameworkException
```

The JPO:method is provided a single args[] array containing a packed Map with these inputs:

- requestMap--Contains the request parameters passed to emxTable.jsp or emxIndentedTable.jsp stored as a map with request parameter names as keys. Also contains objectId, relId, etc.
- columnMap--Details of the dynamic column.
- ObjectList--A MapList (derived from ArrayList) where each entry is a map with details about each objectid, connectionid, and so on.

The JPO:method returns a list with each entry corresponding to a column to add to the table. The columnMap matches the format returned by the UICache component for static columns defined in the administrative table. Data is stored as key/value pairs with settings stored as an additional map.

Each entry in the columnMap list is a map with this structure:

Key	Description	Example Value
name	The name of the column	Person List
label	The text to display as the column header.	
href	The URL to execute when the column data is clicked	\${COMMON_DIR}/emxTree.jsp \${SUITE_DIR}/emxEditPartDialog.jsp
expression_businessobject	The select expression used to obtain the data for the dynamic columns and is applied to the business object specified in the Applies To administrative parameter.	type name \$<attribute[attribute_FindNumber].value>
expression_relationship	The select expression used to obtain the data for the dynamic columns and is applied to the relationship specified in the Applies To administrative parameter.	\$<attribute[attribute_FindNumber].value> \$<attribute[attribute_Qty].value>
sorttype	Defines how to sort the dynamic columns. If you specify other, you also must specify a value for either the Sort Program or Sort Type setting for the column.	numeric alpha (default) other
range	Use if the column needs to have a browse button to call a chooser/custom page for the user to select a value.	\${COMMON_DIR}/emxTypeChooser.jsp? InclusionList=type_Part

alt	For table columns defined with <code>Column Type = Icon</code> setting, the alternate text to display.	<text string>
settings	<p>Key/value pairs of settings for the dynamic columns. The key of the map is the setting name. See Parameters for Toolbar Link Command Objects for a list of the supported settings. Typical settings used for dynamic commands include:</p> <ul style="list-style-type: none"> Editable Column Type program Function Input Type Registered Suite Window Height Access Expression Access Program Access Function 	



Guidelines for Adding Dynamic Columns to a Table

You can follow these general guidelines to include a dynamic column in a table:

1. In Business Modeler, create a new table object.
2. Configure the static columns for the table.
3. Configure the dynamic column for the table. Only one column is defined as dynamic, although the end-result could be any number of columns added to the table. At minimum, include these settings:
 - Column Type = Dynamic
 - Dynamic Column Function = method_name
 - Dynamic Column Program = JPO_name
 - Registered Suite = suite_name
4. Complete the steps required to define a table as defined in the *Business Modeler Guide*.

Any settings defined for the dynamic column apply to all columns added by the JPO:method, unless specifically overridden by the returned MapList.



Sample JPO Code

This code sample retrieves dynamic columns of companies.

```
Public static List getCompanyColumns(Context context, String[] args) throws FrameworkException
{
    // unpack args array to get input map
    MapList programMap = (HashMap) JPO.unpackArgs(args);
    // get request information
    MapList paramMap = (String) programMap.get("requestMap");
    //get dynamic column information
    MapList columnMap = (String) programMap.get("columnMap");
    // get object list
    MapList ObjectList= (String) programMap.get("ObjectList");
    //Define a new MapList to return.
    MapList columnMapList = new MapList();
    // User defined code to populate company list
    // for each company
    for( int i=0; i<companyList.size(); i++){
        // create a column map to be returned.
        Map colMap = new HashMap();
        HashMap settingsMap = new HashMap();
        // Set information of Column Settings in settingsMap
        settingsMap.put("Column Type", "program");
        // Add the column map to the columnMapList
        columnMapList.add(settingsMap);
    }
}
```

```
settingsMap.put("program", "emxUtil");
settingsMap.put("function", "getCompanyColumnValues");
settingsMap.put("Registered Suite", "Framework");
// set column information
colMap.put("name", companyList.get(i).getName() );
colMap.put("label", companyList.get(i).getName() );
colMap.put("settings", settingsMap);
//user added key value pair.
colMap.put("Company Id", companyList.get(i).getObjectId());
columnMapList.put(colMap);
}
// return final list
return columnMapList;
}
```

Table Column Calculations

You can configure tables to provide calculations. Table calculations are available dynamically--using the Table Calculations Options button in the toolbar--or you can configure them to automatically provide calculations for columns and objects.

For information on dynamic table calculations, see "Performing Table Calculations" in the *Application Exchange Framework User's Guide*.

The following topics are discussed:

- [About Table Calculations](#)
- [Table Item Counts](#)
- [Types of Calculations](#)
- [Calculation Settings](#)
- [Decimal Precision](#)

About Table Calculations

Depending on how you define the pagination setting, calculations are given per page, or for all items in the table. The calculation section header shows a count of the number of objects displayed on the page or in the table. You can use column values to show sum, average, maximum, minimum, median, and standard deviation values. You can apply one or more calculation types to each column. The results display at the bottom of the table.

- You can perform calculations only on numerical values. To perform calculations on a column of string attributes that have numerical values, set format=numeric in the table column.
- Null or empty rows are considered as a value of zero. This allows you to perform calculations on data sets that may be missing values for some rows.
- You can perform more than one calculation on the same column. For example, a column may have a sum and an average specified for it. This results in two rows in the calculation section, one for each.
- Place the sum calculation as the first calculation in the section at the bottom of the page. This prevents confusion about what the total represents.
- Page calculations are optional. You can configure it so that only overall calculations are performed and shown them only on the last page of the listing. By default, page level calculations are disabled.
- By default, the Table Calculations Options button displays in the toolbar for read-only (view mode) tables any time the table has numeric columns. You can hide the button by passing `calculations = false` to emxTable.jsp.

If the column contains data for an attribute configured with a dimension, Business Process Services uses the normalized values (the values stored in the database) to perform and display the calculations.



Table Item Counts

All configurable tables show a count of the number of objects used in the calculations in the calculation section header. For page calculations, it shows the number of items shown on the current page. If table pagination is off, then it shows the count of all objects across all pages. This clarifies the scope of the calculations displayed on the page. For example:

- table page 1 has 10 objects
- table page 2 has 10 objects
- table page 3 has 5 objects

If pagination is on and you are looking at page 2, then the item count is displayed as **Page (10 items)**. If pagination is off, then the item count is displayed as **All (25 items)** in the final calculation section. To enable page level calculations, set the property `emxFramework.ShowPageCalculations` to true in emxSystem.properties. This setting is false by default.



Types of Calculations

All configurable tables can provide calculations on columns within the table. This includes the sum of values or the average of all values in the column. You can also show the maximum, minimum, median, or standard deviation of the column.

You can apply calculations to any column in the table that contains numeric-only data except for the first column, which typically contains the object name. To perform calculations on a column of string attributes that have numerical values, you must set format=numeric in the table column. The calculation is performed only for the displayed data. For example:

- table page 1 has objects 1 through 10
- table page 2 has objects 11 through 20
- table page 3 has objects 21 through 25

If pagination is on and you are looking at page 2, then the calculation is applied to objects 11 through 20 only. If pagination is off, then the calculation is applied to all objects (1 through 25).

The following rules apply to page level calculations:

Rule	Description
Show on every page if paginated	If the listing is paginated, page calculations appear on every page.
Disable if pagination is disabled	Page calculations will <i>not</i> appear if pagination is disabled because they would be redundant with the final calculations.
Disable if only one page	Page calculations will <i>not</i> be shown in a paginated view if there is only a single page to be displayed.

Final calculations use all the values for all pages. This section always displays if any calculation settings are enabled, and appear on the last page of the listing. The following example shows a page layout that includes several page level calculations.

The screenshot shows a software interface titled "FT User Defined Configurable Table : multiColumnSort=true". The main area is a table titled "Calculations" with the following data:

Name	Type	Value (Real)	Value (Integer)	Value (String)	Sum
Obj_2_L2	AEFQE_SBCalc	20.77777777	-20	2	2.77778
Obj_2_L3	AEFQE_SBCalc	20.77777777	-20	2	12.77778
Obj_3_L0	AEFQE_SBCalc	30.55555555	-30	3	23.55556
Organisation_1	AEFQE_SBCalc	30.55555555	-30	3	23.55556
Organisation_2	AEFQE_SBCalc	30.55555555	-30	3	23.55556
Total			-480		59.11116
Average			-24		
Maximum			-10		
Minimum			-30		
Median			-30		
Standard Deviation			8.20783		

Below the table, there are summary statistics: Total (-480), Average (-24), Maximum (-10), Minimum (-30), Median (-30), and Standard Deviation (8.20783). The bottom of the interface shows "20 objects".

The following shows an example of both the page level calculations and the final calculations shown at the end of the table.

Table With Preconfigured Calculation Settings

<input type="checkbox"/> Test_TCC_A13	1	AEEQE_TableCharCalcType	Saturday, February 26, 2011	0.5	-1	-4
<input type="checkbox"/> Test_TCC_A14	1	AEEQE_TableCharCalcType	Saturday, February 26, 2011	999.999	-9991	4
<input type="checkbox"/> Test_TCC_A15	1	AEEQE_TableCharCalcType	Saturday, February 26, 2011	90990.123234238	-9991	91919
<input type="checkbox"/> Test_TCC_A2	1	AEEQE_TableCharCalcType	Saturday, February 26, 2011	12.828183	-999	-10
Page (8 items)						
Total				184107.290913305		163835
Average						22979.75
Maximum						91919
Minimum						-9991
Median				555.35854		5405
Standard Deviation				41958.27263		5215.19339
All (15 items)						
Total				275227.459652374		275677
Average						15370.46667
Maximum						91919
Minimum						-9991
Median				112.71828183		1990.0
Standard Deviation				37563.11622		4772.06612

Calculation Settings

You can define the following settings for any numerical column in the table. Labels display in the first column of the listing that is not a checkbox, a radio button, or an icon image. The label column spans the remaining columns up to the first column that contains a calculation. The labels listed in the following table display by default with each of the calculation types.

Setting Name	Values	Description	Label
Calculate Sum	true false	When True, displays the total of the column values.	Total
Calculate Average	true false	When True, displays the Average of the column values. This is calculated using the total of the values in the column divided by number of rows. See Decimal Precision regarding the rounding of this calculated value.	Average
Calculate Maximum	true false	When True, shows the largest number of the column values.	Maximum
Calculate Minimum	true false	When True, shows the smallest number of the column values.	Minimum
Calculate Median	true false	When True, displays the middle number of the column values. If there is even number of values, the average of the two middle values displays. See Decimal Precision for information about the rounding of this calculated value. The median value for the odd number of rows is not rounded since no calculation is done.	Median
Calculate Standard Deviation	true false	When True, displays the standard deviation of the column values. The formula used for standard deviation is: $sd = \text{square root} [\sum(x - \bar{x})^2 / (N-1)]$ where N is the total number of elements. \bar{x} is the mean of the column values See Decimal Precision regarding the rounding of this calculated value.	Standard Deviation

Decimal Precision

For total, maximum and minimum table column calculations, the system does not round the value. Average, median and standard deviation calculations involve division of numbers so rounding is done when necessary. In `emxSystem.properties`, the property `emxFramework.TableCalculations.DecimalPrecision` defines how many digits follow the decimal point when rounding. By default, the calculated value is rounded to 5 digits after the decimal point if the result contains more than 5 digits after the decimal point. For example, if the calculated value is 25.1234562, then the value would be rounded to 25.12346.

Units of Measure for Columns

If you want to implement the units of measure (dimension) in a custom page, you can use Business Process Services's tag library.

The following topics are discussed:

- [Tag Lib Parameters](#)
- [Example Using the UOMAttribute Tag in a Create JSP](#)
- [Example Using the UOMAttribute Tag in a Processing Page](#)
- [Example Using the UOMAttribute Tag in a View Page](#)
- [Example Using the UOMAttribute Tag in an Edit Page](#)

Tag Lib Parameters

When using the taglib, the UOMAttribute class should only be used with attributes that are configured with a dimension. If no dimension is configured for an attribute, then the "asentered" value is used regardless of the specified parameters.

Use the `UOMAttribute` tag to display the attribute value with the unit of measure.

The format for the UOMAttribute tag is:

```
<framework:UOMAttribute  
objectID=<object ID>  
name=<attribute name>  
mode=<view / edit>  
format=<asentered/metric/english>  
/>
```

This table defines the parameters shown above:

Tag Lib Parameter	Description	Possible/Default Values
objectID	Object ID of the object that the attribute is associated with. If the ObjectID is not provided, the attribute value will be shown as empty and the units defined in the dimension will be listed in the drop-down box. Not applicable in a create dialog page.	ObjectID of the business object
name	Name of the attribute configured with a dimension	Weight
mode	Defines whether or not the attribute value is displayed as read-only text or as an editable textbox/combobox field.	View (default) Edit
format	Converts the normalized value stored in the database to the defined format for display purposes. The preference units are the units selected by the user as their preference. When the mode=Edit, the asentered format is used even if the format is set to a different value. Not applicable in a create or edit dialog page.	asentered (default) english metric preference



Example Using the UOMAttribute Tag in a Create JSP

In a create dialog JSP page, this code specifies the unit of measure for an attribute:

```
<framework:UOMAttribute name="AttributeName" mode="Edit"/>
```

`AttributeName` can be the symbolic or actual name of the attribute that is configured with a dimension. If you use the symbolic name, the code retrieves the attribute's actual name and uses that for further processing. This example shows how to use the tag lib to define a field:

```
<tr>  
  <td class="label" width=150>Weight</td>  
  <td class="inputField" colspan="1">  
    <framework:UOMAttribute name="Weight" mode="edit"/>  
    <td>  
  </tr>
```

The result of this code would look like this:



The text box displays the default attribute value. If no default is defined, then the text box is empty. The combo box contains the list of defined units for the dimension, with the default units (kg in this example) displayed.

The name of the text box is `AttributeName` (as defined in the tag lib call). The processing page uses this name to retrieve the value entered by the user. The name of the combo box is `units_AttributeName` and can be used by the processing page to retrieve the units specified by the user.

To define a custom name (other than the attribute name) for the text field and combo box, include the `fieldName` attribute in the tag lib call. In this case, the text box name is `fieldName` and the units combo box name is `units_fieldName`. For example:

```
<tr>
<td class="label" width=150>Weight</td>
<td class="inputField" colspan="1">
    <framework:UOMAttribute name="Weight" mode="edit"
    fieldName="Part Weight"/>
<td>
</tr>
```

In this example, the text box name is "Part Weight", and the combo box name is "units_Part Weight".

Example Using the UOMAttribute Tag in a Processing Page

In a processing page, the Text box value = `emxGetParameter(request, "AttributeName")` and the Combo box value = `emxGetParameter(request, "units_AttributeName")`. The values should be concatenated as follows:

Text box value + " " + Combo box value

For example:

```
String sWeight = emxGetParameter(request, "Weight");
String sWeightUnit = emxGetParameter(request, "units_Weight");
sWeight = sWeight + " " + sWeightUnit;
//Use sWeight as UOM attribute value for processing purpose
```

Example Using the UOMAttribute Tag in a View Page

When displaying an attribute configured with a dimension in a view page, follow this example:

```
<framework:UOMAttribute objectId = "Object Id"
name="AttributeName" mode="view"/>
```

`AttributeName` can be the symbolic or actual name of the attribute that is configured with a dimension. If you use the symbolic name, the code retrieves the attribute's actual name and uses that for further processing. This example shows how to use the tag lib to define a field:

```
<tr>
    <td class="label" width="150">Weight</td>
    <td class="field">
        <framework:UOMAttribute name="Weight" objectId=<%=partId%>
        mode="View"/>
    </td>
</tr>
```

The view page displays the above code as:

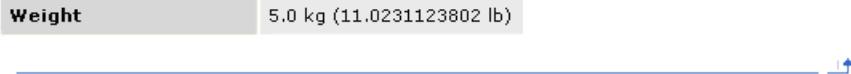


The `fieldName` attribute is not applicable on a view page.

The tag lib uses the As Entered value/units in the view page. To display the English or Metric system values, or the user's preference, you must include the format tag lib attribute as follows:

```
<tr>
    <td class="label" width="150">Weight</td>
    <td class="field">
        <framework:UOMAttribute name="Weight" objectId=<%=partId%>
        mode="View" format="English"/>
    </td>
</tr>
```

The page displays the As Entered value, followed by the format-specified value:



Weight 5.0 kg (11.0231123802 lb) ↑

Example Using the UOMAttribute Tag in an Edit Page

You can use the tag lib in an edit page:

```
<framework:UOMAttribute objectId = "Object Id"  
name="AttributeName" mode="Edit"/>
```

`AttributeName` can be the symbolic or actual name of the attribute that is configured with a dimension. If you use the symbolic name, the code retrieves the attribute's actual name and uses that for further processing. This example shows how to use the tag lib to define a field:

```
<tr>  
    <td class="label" width="150">Weight</td>  
    <td class="inputField">  
        <framework:UOMAttribute name="Weight" objectId="<%partId%>"  
mode="Edit"/>  
    </td>  
</tr>
```

This field on the edit dialog page looks like this:



Weight 5.0 kg ▾

The edit page always displays the As Entered value/units, regardless of the user preferences.

The name of the text box is `AttributeName` (as defined in the tag lib call). The processing page uses this name to retrieve the value entered by the user. The name of the combo box is `units_AttributeName` and can be used by the processing page to retrieve the units specified by the user.

To define a custom name (other than the attribute name) for the text field and combo box, include the `fieldName` attribute in the tag lib call. In this case, the text box name is `fieldName` and the units combo box name is `units(fieldName)`. For example:

```
<tr>  
    <td class="label" width="150">Weight</td>  
    <td class="inputField">  
        <framework:UOMAttribute name="Weight" objectId="<%partId%>"  
mode="Edit" fieldName = "Part Weight"/>  
    </td>  
</tr>
```

In this example, the text box name is "Part Weight", and the combo box name is "units_Part Weight".

Improving Performance of Table Columns

This section explains how to improve the performance of a dynamic user interface table column whose Column Type is set to program or programHTMLOutput.

Both Column Type settings use a method in a Java Program Object (JPO) to obtain the column data.

The input to the JPO method that produces the column data must be a HashMap, which contains a MapList (list of object ids) and a HashMap (request parameters). Most table columns need to get a specific set of data for each object, do some processing on the data, and then add the processed data to the result set.

To accomplish this with the minimum number of calls to the database, use the Studio Customization Toolkit method `BusinessObject.getSelectBusinessObjectData(...)`. Use this method, available in Studio Customization Toolkit 9.5.3.0 and higher, instead of the deprecated `context.getSelectBusinessObjectData` method.

`BusinessObject.getSelectBusinessObjectData`

Parameters	context	the context for this request
	oidList	a StringList of OIDs
	selectStmts	a StringList of select statements (name, owner, etc.)
Returns	a complete set of data of type <code>BusinessObjectWithSelectlist</code> , for all the business objects you passed in as a list	

Parse this output data and process it as required. Add it to the resulting column values before returning from the JPO. Using this method, you make only one database call to get the complete set of data required to process the entire column.

Important: DO NOT use the following approach: Looping through the input MapList to get each object ID, then instantiating the business object for each objectID to get the set of data for that object and processing them to add to the column result list. This approach causes the program to access the database as many times as the number of rows in the column. For example, if there are 50 rows, the program makes 50 database calls to get the data required for this one column.

To see a sample JPO program that uses the `getSelectBusinessObjectData` method, refer to the next section.

Sample JPO for Getting Table Column Values

The following is a sample Java Program Object (JPO) that retrieves the current vault for each object in the list in table rows. You can use this kind of JPO when the Column Type setting for a table column is set to `program` or `programHTMLOutput`. The method `getVault` in this program processes the object list and returns the vault values.

For recommendations on improving the performance of programmatic table columns, see [Improving Performance of Table Columns](#).

The vault names can be obtained by configuring the table column as `businessobject select expression vault` instead of defining the Column Type=`program`. The vault example is used for simplicity and to illustrate the steps involved in writing the JPO for a table column.

```
/*
 * emxTableProgram
 *
 * Copyright (c) 1992-2002 MatrixOne, Inc.
 *
 * All Rights Reserved.
 * This program contains proprietary and trade secret
information of
 * MatrixOne, Inc. Copyright notice is precautionary only and
does
 * not evidence any actual or intended publication of such
program.
 *
 * static const char RCSID[] = $Id: Exp $
*/
import matrix.db.*;
import matrix.util.*;
import java.io.*;
import java.util.*;
import com.matrixone.framework.beans.*;
import com.matrixone.framework.util.*;
import com.matrixone.framework.ui.*;
/**
```

```

* @version AEF 9.5.0.0 - Copyright (c) 2002, MatrixOne, Inc.
*/
public class ${CLASSNAME}
{
    /**
     *
     * @param context the Matrix <code>Context</code> object
     * @param args holds no arguments
     * @throws Exception if the operation fails
     * @since AEF 9.5.0.0
     * @grade 0
     */
    public ${CLASSNAME} (Context context, String[] args)
        throws Exception
    {
        if (!context.isConnected())
            throw new Exception("not supported on desktop
client");
    }
    /**
     * This method is executed if a specific method is not
specified.
     *
     * @param context the Matrix <code>Context</code> object
     * @param args holds no arguments
     * @returns nothing
     * @throws Exception if the operation fails
     * @since AEF 9.5.0.0
     */
    public int mxMain(Context context, String[] args)
        throws Exception
    {
        if (!context.isConnected())
            throw new Exception("not supported on desktop
client");
        return 0;
    }

    /**
     * get Vault for the objects.
     *
     * @param context the Matrix <code>Context</code> object
     * @param args holds the following input arguments:
     *          0 - HashMap programMap
     * @returns vector of Vault names
     * @throws Exception if the operation fails
     * @since AEF 9.5.0.0
     */
    public static Vector getVault(Context context, String[]
args)
        throws Exception
    {
        HashMap programMap = (HashMap) JPO.unpackArgs(args);

        MapList relBusObjPageList =
(MapList)programMap.get("objectList");
        HashMap paramMap =
(HashMap)programMap.get("paramList");

        Vector vaultList = new Vector();
        StringList listSelect = new StringList(1);
        listSelect.addElement("vault");
        String objIdArray[] = new
String[relBusObjPageList.size()];

        if ( relBusObjPageList != null)

```

```
{  
    for (int i = 0; i < relBusObjPageList.size(); i++)  
        objIdArray[i] =  
(String)((HashMap)relBusObjPageList.get(i)).get("id");  
    // get the vault for the object  
    BusinessObjectWithSelectList vaultSelectList = null;  
    vaultSelectList =  
BusinessObject.getSelectBusinessObjectData(objIdArray,  
listSelect);  
    for (int i = 0; i < relBusObjPageList.size(); i++)  
    {  
        String vaultName =  
vaultSelectList.getElement(i).getSelectData("vault");  
        vaultList.add(vaultName);  
    }  
    return vaultList;  
}  
}
```

Additional URL Parameters for Tables

In addition to directly defining the URL parameters to pass to emxTable.jsp or emxTableEdit.jsp, you can use the `appendURL` parameter to retrieve additional URL parameters from the properties file for the ENOVIA application.

The `appendURL` parameter takes a value in this format:

`<KEY>|SuiteKey`

Where the `KEY` is a component of a property in the properties file of the specified `SuiteKey`. For example, `appendURL=Effectivity|EngineeringCentral`

The system uses this value, plus the UI component where the `appendURL` is defined (in this case a table), to fetch the list of additional URL parameters to pass to the UI component. The above example corresponds to this property in the `emxEngineeringCentral.properties` file:

```
emxEngineeringCentral.Table.Effectivity =  
appendColumns=AdditionalColumnsTable&toolbar={$ORIGINAL},effectivitytoolbar
```

In this example, 2 URL parameters are being returned to the Table page that specified the `appendURL` parameter: `appendColumns` and `toolbar`. The `$_ORIGINAL` macro is used to replace the URL parameter in the original URL string with the one specified in this property. In this example, the toolbar defined in the URL string would be replaced by the one named `effectivitytoolbar`.

As shown by the above example, the `emxSUITENAME.properties` file must contain the required key specified in this format:

`emx<SuiteKey>.<ComponentKey>.<KEY>=VALUE`

where:

- `emx<SuiteKey>` is the application's suite key, such as `emxEngineeringCentral`
- `<ComponentKey>` is the type of UI component requesting the additional URL parameters and is one of these values:
 - Create
 - Form
 - Table
 - StructureBrowser
- `<KEY>` is the specific key being looked up
- `VALUE` is an ampersand-separated list of URL parameters

URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp

This table lists the parameters that you can use with emxTable.jsp and emxTableEdit. You can add these parameters to the href parameter for the component that calls the table.

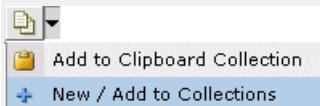
For example, when you specify the emxTable.jsp to call from a tree category, you can add these parameters to the href parameter for the command object.

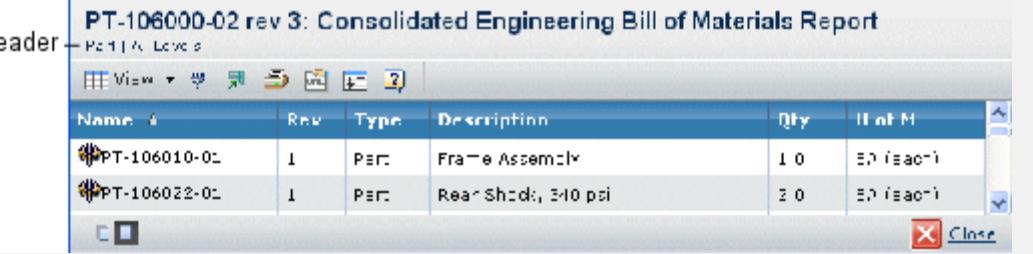
Parameters that are specific to view mode tables or edit mode tables are denoted as (view mode only) or (edit mode only).

Parameter	Description	Accepted Input Values
appendColumns	The name of the table that defines columns that you want to include in the table being defined.	
autoFilter	Determines whether the filter tool  displays in the toolbar, and overrides the Auto Filter setting on a table or structure browser column.	true (default)--The Filter tool displays in the page toolbar. false--The Filter tool does not display in the page toolbar.
CancelButton (view mode only)	Displays the Cancel link in the table footer frame. When a user clicks Cancel, the whole browser window closes. You can only use the Cancel link for a table page opened in new window. You can configure the label for the Cancel link using the CancelLabel parameter. If you do not pass the CancelLabel, the default label "Cancel" is used.	true--Shows the Cancel link. Use only for tables displayed in popup windows. false (Default)--Does not show the link.
CancelLabel (view mode only)	Defines the label for the Cancel link, if you want something other than "Cancel".	Any static text or string resource id. CancelLabel=emxEngineeringCentral.Common.Cancel
chart (view mode only)	Specifies whether the chart options icon displays in the table toolbar.	true (default)--Show the chart options icon in the table toolbar if the table has one or more numerical columns. No icon is shown if the table does not have any numerical columns. false--Do not show the chart options icon in the table toolbar, even if the table has numerical columns.
customize	Enables (true) or disables (false) the ability for users to create customized versions of the page. This parameter overrides the emxFramework.UITable.Customization system property, and has no effect on other table or structure browser pages. If not provided, the value specified for <code>emxFramework.UITable.Customization</code> is used. All access controls defined for the system table are retained in the customized table. If a column does not have a label or alt value defined (and is not a file, checkbox, icon, image, or separator), the column name shows in the user's selection list, however, this name cannot be internationalized.	true false
disableSorting (view mode only)	In the table component, by default all the columns are sortable, unless a column has a setting 'Sortable = false'. If the user clicks on the header of the sortable column, then the objects are sorted based on that column. To prevent users from sorting columns, pass the parameter 'disableSorting=true' to emxTable.jsp.	true false (default)
editLink (view mode only)	Use to display the Edit toolbar item on the table page. The Edit link is automatically configured to call the editable version of the current table using emxTableEdit.jsp.	true false (default)
Export (view mode only)	Shows or hides the Export  tool on the page toolbar. When users click the tool, the system exports the table data to a file in the user's preferred format in their preferred format: CSV, HTML, or Text. Users choose their preferred format using the Preferences tool in the global toolbar. For information on configuring the export preference, see About Preferences . For instructions on using the preference page, click Help from the preference page. If a column includes an attribute configured with a dimension, the exported data shows the value/units as displayed in the table, not the normalized values. If the user chooses CSV or Text for their export format, the export does not include Icon column type and programHTMLOutput columns. To get the column values exported when the column type is "programHTMLOutput," you must also set "Export= true" for the column.	true (Default)--The Export tool displays, allowing users to export the table data. false--The tool does not display.
FilterFramePage (view mode only)	Specifies the name of the JSP used in the custom filter frame (between the header and body). You can use the directory macros to refer to the location of the page. The path of the page is relative from the ematrix/common directory. For details on implementing the custom filter, see Implementing a Custom Filter .	\${SUITE_DIR}/emxTeamProjectTableFilter.jsp? \${ROOT_DIR}/emxPartECOSatusFiletr.jsp? \${COMMON_DIR}/emxCommonStateFilter.jsp? emxCommonStateFilter.jsp
FilterFrameSize	Controls the size of the filter frame in pixels. The value should be at least 40 (the default).	40 (default)

(view mode only)		80 160
findMxLink	When true, show the mxLink icon/command button on the toolbar, which opens a search dialog box.	true (default) false
header	The content of the heading that appears at the top of the table page.	Any alphanumeric text or a string resource ID. header=Buyer Desk header=emxQuoteCentral.AssignedPackages.AssignedPackages
headerRepeat	Specifies how often the header row should repeat. A repeating header row makes it easier for users to identify the content of each column when scrolling through a long table. If the parameter is not used, the header row repeats every 15 rows (or whatever number is assigned to emxNavigator.UITable.HeaderRepeat in emxSystem.properties).	Any positive integer. Default is 15. For example, 11 would mean the header row is repeated every 11 rows. To turn off headerRepeat, pass in 0 as the value for the parameter.
HelpMarker	Specifies the name of the help marker to call for context-sensitive help. For information about implementing help, see Implementing Context-Sensitive Help for FrameMaker Source .	String The naming convention for help markers "emxpath" followed by the object or feature and then the action, for example, emxhelproutecreate and emxhelpprojectedit. The marker is all lowercase with no spaces.
inquiry	Used when there is no <code>program</code> parameter passed in to emxTable.jsp or emxTableEdit.jsp. Specifies the inquiry administrative object to retrieve the business objects to include in the table and any inquiries for the table page filter list. The emxTable.jsp uses the first inquiry object to display the list in the table on page load. It uses all inquiry objects for the table filter list in the header. emxTableEdit.jsp does not support multiple inquiries. If multiple inquiries are passed with the ',' separator, only the first one is used. It ignores the rest. If there is only one inquiry, the filter list in the header is not displayed. To build the filter list, you must use either JPOs or Inquiry objects. You cannot use both.	For emxTable.jsp, Names of inquiry administrative objects separated by commas. Only one inquiry is allowed for emxTableEdit.jsp. inquiry=SCSBuyerDesk,SCSBuyerDeskAssigned inquiry=ENCAIIParts, ENCReleasedParts
inquiryLabel (view mode only)	Used with the "inquiry" parameter. Specifies the labels for each inquiry in the filter list in the table page header. Each label is associated to the corresponding inquiry administrative object name specified in the "inquiry" parameter. The label may have the actual text or the string resource id for internationalization.	One or more labels (equal to the number of objects specified in the "inquiry" parameter) for the inquiry administrative object names separated by a comma. inquiryLabel=All,Assigned inquiryLabel= emxEngineeringCentral.Common.All, emxEngineeringCentral.Common.Released
jpoAppServerParamList	Allows session data to be passed to a JPO and uses the format: <code>scope:attributeName</code> where scope can be one of these values: <code>application</code> <code>session</code> <code>request</code> and the attribute must be a valid attribute used within the specified scope. The parameter can pass a comma-separated list of scope:attributeName values. The attribute values must be serializable.	application:<attributeName>,session:<attributeName>,request:<attributeName>
launched (view mode only)	When the user clicks the Launch button (from a channel tab), the system passes <code>launched=true</code> to the new popup page. It indicates that the popup is a result of clicking the Launch button. You can change the behavior of the popup page from the normal mode pages using this parameter. In normal mode, pages will not include the launched parameter and default to <code>launched=false</code> .	true false (default)
massPromoteDemote	Overrides the system property <code>emxFramework.Lifecycle.MassPromoteDemote.Enable</code> for the specific page. The default value for this parameter is the value for the above property. The page must also include the State and Type columns. See the <i>Live Collaboration Administrator's Guide</i> for details on setting property values that control the mass promote/demote feature.	true false
massUpdate (edit mode only)	Use to turn on or off mass update controls on the editable table page.	true (default)--Mass update is available for the page. false--Mass update not is available for the page.
multiColumnSort	Enables or disables multiple column sorting on a page. If disabled, the user can still sort by a single column if one is specified in the <code>sortColumnName</code> parameter. When set to false, multiple column sorting is also disabled in any custom tables users' create based on this system table.	true (default) false
objectBased (view mode only)	Use to configure a table page to list items other than business objects. For example, the items such as collections and IconMail messages are not business objects you can list them in a table by passing the URL parameter <code>objectBased=false</code> . When you set <code>objectBased</code> to false, the parameter <code>objectCompare</code> is assumed to be false and	true (default)--The table page lists only business objects. false--The table page lists items that are not business objects.

	<p>the object compare icon does not display.</p> <p>When you set <code>objectBased</code> to false, the JPO that gets the list should follow the same guidelines used for getting the object list. See JPO Guidelines for Getting Object List. Assign the MapList containing the HashMaps with key name "id" to the row identifier value and not the "objectId." The value assigned to "id" is the row identifier for every row in the table. If the table has a check box column (named emxTableRowId), the check box value is assigned "id" value.</p>	
objectCompare (view mode only)	<p>Use to display or hide the object compare icon. When a column contains an attribute configured with a dimension, the comparison is done based on the normalized value, not the displayed value.</p> <p>When either of the following conditions apply, this parameter is assumed to be false and the object compare icon does not display:</p> <ul style="list-style-type: none"> • URL parameter <code>objectBased</code> is set to false. • URL parameter <code>selection</code> is set to single or none. 	true (default)--The object compare icon is displayed. false--The object compare icon is not displayed.
pagination (view mode only)	<p>Specifies the number of rows to show per page. If the rows span more than one page, users can use the Left and Right Arrow buttons or the Page drop-down list to navigate to other pages. For more information on pagination controls, see Table Pagination Controls.</p> <p>If you do not specify this parameter, the system uses the value set in the pagination property in <code>emxSystem.properties</code>. When installed, this property is set to 10 rows per page. If 0 is specified, all objects are listed on one page.</p>	Any number. Use 0 if you do not want to paginate. If 0 is specified, the pagination controls do not include the controls for navigation.
portalMode (view mode only)	<p>Every page configured inside the PowerView includes the parameter <code>portalMode=true</code>, so that the page can differentiate between normal display and portal display.</p> <p>Clicking the Launch button, launches the currently displayed channel tab into a maximized popup window, passing the parameters <code>launched=true</code> and <code>portalMode=false</code> to the new window.</p>	true false (default)
portableTable (view mode only)	<p>Specifies the alternate table to use within a channel tab.</p> <p>Displaying table component listings in half page-width channels, typically cause the user to scroll horizontally to view all columns. However, you can create an alternate table with fewer columns for display specifically within a channel tab. For example, if a standard ECO New Parts table contains 10 columns, you could define an alternate ECO New Parts table containing 4 columns.</p>	Name of alternate table
PrinterFriendly (view mode only)	<p>Specifies whether the table page includes the Printer Friendly tool.</p> <p>You can have the system pass this parameter automatically by entering the Printer Friendly setting for the command object that calls the table page.</p>	true (default) false
program	<p>Used only when there you do not pass the <code>inquiry</code> parameter to <code>emxTable.jsp</code> or <code>emxTableEdit.jsp</code>.</p> <p>This parameter provides an alternative to the inquiry object. The <code>program</code> approach uses a JPO program object to fetch the object list to display.</p> <p>For <code>emxTable.jsp</code>, the parameter value can have one or more sets of values separated by a comma ",".</p> <p>The <code>emxTableEdit.jsp</code> does not support multiple program parameters. If you pass multiple programs with the ',' separator, it uses only the first and ignores the rest.</p> <p><code>emxTable.jsp</code> uses the first set of values (JPO name and method name) to display the list when the page is first loaded. It uses all the sets for the table filter list in the header.</p> <p>If there is one program, the filter in the header does not display.</p> <p>To build the filter list, use either JPOs or Inquiry objects. You cannot use both.</p>	<p>This parameter is assigned to one (or more in view mode) set of values that will form a JPO program name and the method name. The format of the parameter value is: <code>program=<JPO program name>:<JPO method name></code></p> <p>The program name and the method name are separated by a colon ":".</p> <p><code>program=emxTableBuyerDesk:getBuyerDesk,emxTableBuyerDesk:getAssignedBuyerDesk</code></p> <p>For guidelines on writing a JPO for getting a list of business objects, see JPO Guidelines for Getting Object List. For a sample of a JPO that gets a list of business objects, see Sample JPO for Getting List of Objects.</p>
programLabel (view mode only)	<p>Use with the parameter "program" defined above. Specifies the labels for each program in the filter list in the table page header.</p> <p>You can assign one or more labels (equal to number of "program" values) separated by a comma ",".</p> <p>Each label is associated to the corresponding JPO object and method name specified in "program" parameter.</p> <p>The label can contain the actual text or the string resource id for internationalization.</p>	<code>programLabel =All,Assigned</code> <code>programLabel = emxEngineeringCentral.Common.All,</code> <code>emxEngineeringCentral.Common.Released</code>
rememberSelection (view mode only)	<p>Controls whether the system remembers the items a user selects on one page when the user navigates to another page in the table. The <code>emxSystem.properties</code> key <code>emxNavigator.UITableView.RememberSelection</code> sets the default for table pages. Use this parameter to override the default for a single page.</p> <p>For more information, see Table Pagination Controls.</p>	true--Selected items are remembered across the table's pages. false--Selected items are not remembered across pages.
selection (view mode only)	Controls whether the table page adds a column of check boxes or radio buttons in the left-most column of the table.	multiple--Users can select more than one row in the table, and the system displays a check box in the left column of each row. There are no access restrictions for the check

	<p>When this parameter is set to <code>single</code> or <code>none</code>, the parameter <code>objectCompare</code> is assumed to be false and the object compare icon does not display.</p>	<p>boxes. If you pass this parameter, it overrides any check box column added to the table administrative object. For more information, see Column Values as Check Boxes.</p> <p><code>single</code>--Users can select one row in the table. A radio button displays in the left column of each row.</p> <p><code>none</code>--A selection column is not added to the table page. (You can still display a check box column by adding the column to the table administrative object.)</p>
showClipboard	<p>This menu is enabled by default and adds the  to the page toolbar. The icon acts as a pull-down menu:</p>  <p>If the user clicks , selected objects are added to the clipboard collection for that user. If the user clicks the arrow, the user can select the New/Add to Collections or Add to Clipboard Collection command.</p> <p>Set the value for this parameter to false to disable this feature.</p> <p>For File Summary Pages for ENOVIA products, the default value for this parameter is false.</p>	true (default) false
showPageURLIcon	<p>When true,  shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false,  does not show in the toolbar.</p> <p>The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in <code>emxSystem.properties</code>.</p>	true false
showRMB	Enables or disables the right-click menus on the page. When set to false, all right-click menus for that page are disabled.	true (default) false
showTabHeader	<p>Applies only in Portal mode (when the page is displayed within a PowerView), enables or disables the page header.</p> <p>If false, the header text is not displayed in the PowerView tab.</p> <p>If true, the header text shows in the tab.</p>	true false (default)
sortColumnName	<p>Specifies the sort columns when the page first loads. If no column is specified, the rows display in the order they are retrieved from the database.</p> <p>Can be a comma-separated list of up to 3 column names. The page sorts by the first provided column, then by the second, then by the third.</p> <p>Used as the default Sort by settings in the Customize Table View dialog box.</p> <p>Columns defined with <code>sortType=other</code> setting cannot be used with <code>multiColumnSort</code>.</p>	Name of table column. column1,column2,column3
sortDirection	<p>Defines the sort order for the columns specified in the <code>sortColumnName</code> parameter. If a single value is passed, it applies to all columns, or you can pass a comma-separated list of values matching the number of columns in the <code>sortColumnName</code> parameter.</p> <p>Used as the default sort directions in the Customize Table View dialog box.</p>	ascending (default)--Sort a to z or 0 to n. descending--Sort z to a or n to 0. ascending,descending,ascending
stopOOTBRefresh	After opening the Edit view of a table, the value for this parameter determines if the the View mode of the table is refreshed based on the user edits. When true, the table is refreshed; when false, it is not refreshed.	true (default) false
Style (view mode only)	<p>Determines the cascading style sheet that defines the layout and color for headers and footers on the table page or popup dialog.</p> <p>If the style parameter is not specified, the system uses the <code>list</code> style from the Configuring Styles style sheet. To use a different style sheet, change the following property in <code>emxSystem.properties</code>:</p> <pre>emxNavigator.UITable.Style.List = styles/emxUIList.css</pre> <p>If you specify <code>style = dialog</code>, the system uses the Configuring Styles style sheet. To use a different style sheet, change the following property in <code>emxSystem.properties</code>:</p> <pre>emxNavigator.UITable.Style.Dialog = styles/emxUISearch.css</pre> <p>If you specify <code>style = PrinterFriendly</code>, the system uses the Configuring Styles sheet. This is hard-coded into the <code>emxTableReportView.jsp</code> page (the page that displays the printer-friendly page).</p>	list (default)--Use for table pages displayed in the content frame of <code>emxNavigator.jsp</code> . dialog--Use for table pages displayed in popup windows, such as search result pages. PrinterFriendly--Use for displaying a page in printer-friendly mode directly from a command link, instead of the Printer Friendly icon in a list page.
subHeader	Creates a subHeader below the main header in the table header frame.	The value can be static text or a string resource id. For example: <code>subHeader= emxEngineeringCentral.Common.BOMLevel</code> <code>subHeader=Bill of Material Level 1</code>

		The value can also include macros such as \${type} \${revision}.
SubmitURL (view mode only)	<p>Displays the Submit link in the table footer frame. The value can be a valid JSP page that executes when the user clicks Submit.</p> <p>You can configure the label for the Submit link using the SubmitLabel parameter. If SubmitLabel is not passed, the system uses the default label "Submit".</p>	Any JSP. The path can include any directory macro. SubmitURL=\${SUITE_DIR}/emxBlank.jsp
SubmitLabel (view mode only)	Defines the label for the Submit link.	Any static text or string resource id. SubmitLabel=emxEngineeringCentral.Button.Next
table	Specifies the table object for presenting this targeted page. In view mode, the table object has all the information needed to display the table, including the columns to present. You must use the actual table name; this parameter does not support symbolic names.	Name of table administrative object. For example: table=SCSBuyerDesk table=ENCParts
tableType	If the emxFramework.Table.Type property in emxSystem.properties is set to new, this URL property overrides that setting. The only accepted value is classic. See Table-to-Structure Browser Conversion for details.	classic
targetLocation	When a URL is defined to open in the slide-in frame (the Target Location setting for the component is set to slidein), Business Process Services appends this URL parameter with this value. This parameter supports a custom page where the html for a popup window needs to be different than the html for a slide-in window.	slidein
TipPage (view mode only)	Specifies whether the table page should include the Tip Page tool and call a specific html or jsp when a user clicks the tool. If this setting is not included, the Tip tool is not included on the table page.	Name of a custom html or JSP page, including any path. The starting point for the directory reference is the content directory. For example, to call an html file in ematrix/doc/customcentral and the content directory is ematrix/customcentral, you would add this parameter to the table.jsp: TipPage=../doc/customcentral/tippage.html
toolbar	<p>Defines the name of the toolbar menu to show on the table page. You can pass this to either emxTable.jsp or emxTableEdit.jsp. To use the same toolbar in both a view mode and an editable table, pass the same toolbar name to each.</p> <p>You show the Edit button in the Table component by passing the parameter editLink = true to emxTable.jsp, then passing the editToolbar parameter to emxTable.jsp.</p> <p>If the Edit button in the Table component is a custom command where the URL is a custom URL (for example emxTableEdit.jsp), then the parameter should be appended to the custom URL (emxTableEdit.jsp).</p>	Name of the toolbar: toolbar=SCSBuyerDesktoolbar
TransactionType	Controls whether the table query executes within an Update transaction or Read transaction. Use the Update transaction whenever the code needs to update the database while fetching the object list.	read (default)--The table query does not execute within an Update transaction. update--The table query (inquiry or JPO) executes within an Update transaction.

JPO Guidelines for Tables

This section presents guidelines and samples for writing JPOs that act on tables and columns.

In this section:

- ❑ [JPO Guidelines for Getting Object List](#)
- ❑ [Sample JPO for Getting List of Objects](#)
- ❑ [JPO Guidelines for Updating Column Values](#)

JPO Guidelines for Getting Object List

You can write and configure JPOs to get the list of objects to display in the table page.

A JPO program object for getting an object list must follow these rules:

- The JPO program must define a method with the name specified in the href parameter using the form: program=<program name>:<method name>.
- The method takes an input argument as a HashMap. The HashMap contains the details of the request parameters available to the JSP page. Every parameter is available as a key/value pair in the HashMap. In addition, a parameter called languageStr is available to get the language setting (such as en, fr, ja, etc).
- The method returns a MapList containing the list objects. This MapList contains a list of HashMaps with the values of OIDs and RelIDs. Every HashMap contains the value for OID and RelID belonging to one table row.

This is the structure of each HashMap belonging to a table row:

- Key "id" is assigned to the business object Id (OID) to use in the table row.
- Key "id[connection]" is assigned to the relationship id (RelID) to use (if required) for the table row.

For example:

MapList		
Object	Key	Value
HashMap 1	id	Value
	id[connection]	Value
HashMap 2	id	Value
	id[connection]	Value
...	id	Value
	id[connection]	Value
HashMap n	id	Value
	id[connection]	Value

Here is a JPO function template:

```
public static Object methodName(Context context, String[] args)
throws Exception
{
    HashMap paramMap = (HashMap)JPO.unpackArgs(args);
    MapList objectList = new MapList();
    // Get the required parameter values from "paramMap" - if
    required
    String languageStr = (String)paramMap.get("languageStr");
    String objectId = (String)paramMap.get("objectId");
    String selectedFilter =
    (String)paramMap.get("selectedFilter");
    // Get the object elements - OIDs and RELIDs using Beans or ADK
    methods
    // Build the MapList with List of HashMaps containg OIDs and
    RelIDS
    return objectList;
}
```

Sample JPO for Getting List of Objects

The following is a sample JPO for obtaining the list of objects to display in a table.

```
/*
 * emxTableListJPO
 *
 * Copyright (c) 1992-2002 MatrixOne, Inc.
 *
 * All Rights Reserved.
 * This program contains proprietary and trade secret
information of
 * MatrixOne, Inc. Copyright notice is precautionary only and
does
 * not evidence any actual or intended publication of such
program.
 *
 * static const char RCSID[] = $Id: Exp $
*/
import matrix.db.*;
import matrix.util.*;
import java.io.*;
import java.util.*;
import com.matrixone.framework.beans.*;
import com.matrixone.framework.util.*;
import com.matrixone.framework.ui.*;
/**
 * @version AEF 9.5.0.0 - Copyright (c) 2002, MatrixOne, Inc.
 */
public class ${CLASSNAME}
{
    /**
     *
     * @param context the Matrix <code>Context</code> object
     * @param args holds no arguments
     * @throws Exception if the operation fails
     * @since AEF 9.5.0.0
     * @grade 0
     */
    public ${CLASSNAME} (Context context, String[] args)
        throws Exception
    {
        if (!context.isConnected())
            throw new Exception("not supported on desktop
client");
    }
    /**
     * This method is executed if a specific method is not
specified.
     *
     * @param context the Matrix <code>Context</code> object
     * @param args holds no arguments
     * @returns nothing
     * @throws Exception if the operation fails
     * @since AEF 9.5.0.0
     */
    public int mxMain(Context context, String[] args)
        throws Exception
    {
        if (!context.isConnected())
            throw new Exception("not supported on desktop
client");
        return 0;
    }
}
```

```
/**  
 * Get the list of objects.  
 *  
 * @param context the Matrix <code>Context</code> object  
 * @param args holds the following input arguments:  
 *          0 - objectList MapList  
 * @returns Object of type MapList  
 * @throws Exception if the operation fails  
 * @since AEF 9.5.1.2  
 */  
public MapList getRTSList(Context context, String[] args)  
    throws Exception  
{  
    HashMap paramMap = (HashMap)JPO.unpackArgs(args);  
    String objectId = (String)paramMap.get("objectId");  
    String selectedFilter =  
(String)paramMap.get("selectedFilter");  
    String programList = (String)paramMap.get("program");  
    String filterName = "All";  
    MapList relBusObjPageList = new MapList();  
    SelectList selectListRTS = new SelectList(1);  
Person person = Person.getPerson(context);  
    selectListRTS.add(person.SELECT_ID);  
    relBusObjPageList = person.getOwnedRTSs(context,  
selectListRTS, true, null, null, false, filterName);  
    return relBusObjPageList;  
}  
}
```

JPO Guidelines for Updating Column Values

You can use this JPO approach for obtaining column values when the column is configured with an update program and update function.

Use this approach for a column with these settings:

- Update Program=JPO name
- Update Function=JPO method name

For processing the data, the input parameter may be used by the method. The return type of this method is a Boolean, which is true if the update is successful. The JPO function template looks like this:

```
public static boolean methodName(Context context, String[] args)
throws Exception
{
    HashMap programMap = (HashMap) JPO.unpackArgs(args);
    HashMap paramMap = (String) programMap.get("paramMap");
    HashMap requestMap = (String) programMap.get("requestMap");
    // Get the required parameter values from "paramMap" - as
    required
    String objectId = (String) paramMap.get("objectId ");
    String relId = (String) paramMap.get("relId ");
    String newValue = (String) paramMap.get("New Value");
    String oldValue = (String) paramMap.get("Old Value");
    // get languagestr from requestmap
    String languageStr = (String) requestMap.get("languageStr");
    // Define and add selects if required
    // Process the information to set the field values for the
    current object
    ?.
    return 0;
}
```

See [Improving Performance of Table Columns](#) for a sample JPO for getting table column values.

In the JPOs used for updating the Table component fields, the values for the parameters timeZone, charSet, localeObj are returned as String[] objects. The first element of these string arrays contain the parameter values. And the languageStr parameter is returned as a string.

Custom Filter for a Table

You can define custom filterings using a custom JSP.

In this section:

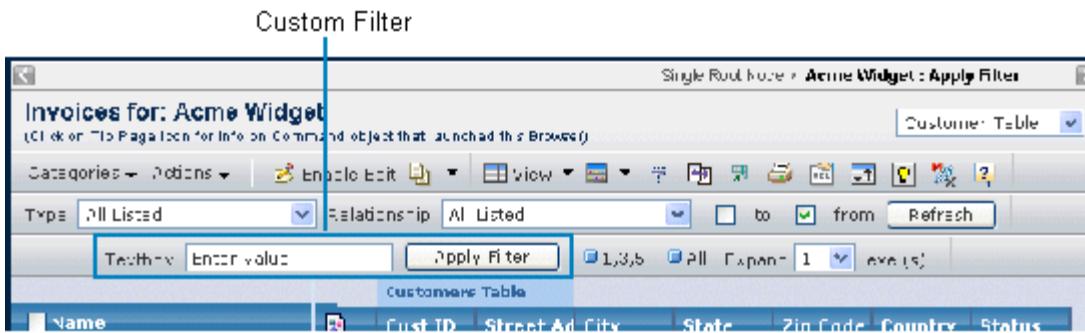
-  [Implementing a Custom Filter](#)
-  [Sample JSPs for Custom Filtering](#)

Implementing a Custom Filter

You can define and implement a custom filtering scheme for a table page.

1. Create a custom JSP page to access the table data and do the filtering based on the user input.

This graphic shows an example of a custom filter implemented on a table page.



The custom JSP page gets an input request parameter "tableID." Read this parameter from the request object. This parameter is the key to obtain the current table data.

2. Using the table ID key, call the table bean methods (described below) to obtain the table data.
3. The table data contains the complete list of objects and other input parameter details. Process the table data to get a filtered list and set the filtered objects list back in the table data using a bean method.
4. Refresh the table display page using a JavaScript API.

The following code is a template that illustrates the custom filtering described above:

```
////////// START Code Template //////////
//Declare the table bean with session scope
<jsp:useBean id="tableBean"
class="com.matrixone.apps.framework.ui.UITable"
scope="session"/>
// Start the JSP code
<%
// Get the table ID from the request
String tableID = Request.getParameter(request, "tableID");
// Obtain the object list for the current table using the
tableID
MapList relBusObjList = tableBean.getObjectList(tableID);
// Obtain the requestMap which contain all the request params to
emxTable.jsp
HashMap requestMap = tableBean.getRequestMap(tableID);
?.
// Process the object List and filter it to create a new MapList
of filtered object List
MapList filteredObjPageList = ?.
?
// Set the filtered object list for the current table using the
tableID
tableBean.setFilteredObjectList(tableID, filteredObjPageList);
// End JSP code and Start the Javascript code to call the
refresh method as below
%>
<script language="JavaScript">
    parent.refreshTableBody();
</script>
////////// END Code Template ///////////
```


Sample JSPs for Custom Filtering

You can create custom JSPs to implement custom filtering.

Two JSP pages might be required to implement the custom filtering within the configurable table as follows:

- The first page creates the user interface to initiate the filter.
- The second page takes the input from the first page and processes the data. It then filters the data and updates the display. Only the first page is assigned as the URL parameter to emxTable.jsp.

Here are samples of both JSPs.

First JSP (for UI): emxTableFilterIncludeSample.jsp

```
<html>
<%@include file = ".../common/emxNavigatorInclude.inc"%>
<jsp:useBean id="formEditBean"
class="com.matrixone.apps.framework.ui.UIForm" scope="session"/>
<%
String tableID = Request.getParameter(request, "tableID");
%>
<head>
<link rel="stylesheet" href="styles/emxUIDefault.css"
type="text/css">
<link rel="stylesheet" href="styles/emxUIList.css" type="text/
css">
<script language="javascript">
    function filterData()
    {
        var filterProcessURL="emxTableFilterIncludeProcess.jsp";
        document.filterIncludeForm.action = filterProcessURL;
        document.filterIncludeForm.target = "listHidden";
        document.filterIncludeForm.submit();
    }
</script>
</head>
<body>
<form name="filterIncludeForm">
<table border="0" cellspacing="2" cellpadding="0" width="100%">
<tr>
<td width="99%">
<table border="0" cellspacing="0" cellpadding="0" width="100%">
<tr>
<td class="pageBorder"></td>
</tr>
</table>
</td></tr>
</table>
<table>
<tr>
<td width="100"><label>State</label></td>
<td class="inputField" ><select name="filterState" id="">
<option value="Preliminary">Preliminary</option>
<option value="Release">Release</option>
<option value="Create" selected >Create</option>
<option value="Obsolete">Obsolete</option>
</select>
</td>
<td width="50"> &ampnbsp </td>
<td width="150"><input type="button" name="btnFilter"
value="Filter..." onclick="javascript:filterData()"></td>
</tr>
</table>
<input type="hidden" name="tableID" value="<%=>
</pre>
```

```

</form>
</body>
</html>
Second JSP (for processing and refreshing): emxTableFilterIncludeSample.jsp
<%@include file = ".../common/emxNavigatorInclude.inc"%>
<jsp:useBean id="tableBean"
class="com.matrixone.apps.framework.ui.UTable"
scope="session"/>
<%
String tableID = Request.getParameter(request, "tableID");
String selectedState = Request.getParameter(request,
"filterState");
// Get business object list from session level Table bean
// HashMap tableData = tableBean.getTableData(tableID);
MapList relBusObjList = tableBean.getObjectList(tableID);
HashMap requestMap = tableBean.getRequestMap(tableID);
MapList filteredObjPageList = new MapList();
StringList busSelects = new StringList();
BusinessObjectWithSelectList busObjws1 = null;
if (relBusObjList != null)
{
    String objIdArray[] = new String[relBusObjList.size()];
    busSelects.add("current");
    for (int i = 0; i < relBusObjList.size(); i++)
        objIdArray[i] =
(String)((HashMap)relBusObjList.get(i)).get("id");
    busObjws1 =
BusinessObject.getSelectBusinessObjectData(context, objIdArray,
busSelects);
    for (int i = 0; i < relBusObjList.size(); i++)
    {
        String currentStatus =
(String)busObjws1.getElement(i).getSelectData("current");
        if (currentStatus != null &&
currentStatus.equals(selectedState))
            filteredObjPageList.add(relBusObjList.get(i));
    }
}
tableBean.setFilteredObjectList(tableID,
filteredObjPageList);
%>
<script language="JavaScript">
    parent.refreshTableBody();
</script>

```

Sorting Programs for Tables

This section describes how to write a JPO with a custom algorithm for sorting a table based on the values in a column.

In this section:

- [Guidelines for Custom Sort JPO](#)
- [Sample Sorting Code](#)
- [BPS Custom Sorting Programs](#)

Guidelines for Custom Sort JPO

You can write a JPO with a custom algorithm for sorting a table based on its column values.

To use this type of JPO, you must set the table column parameter sort type to other, and the table column setting Sort Program must contain the name of the JPO. You can use the multiColumnSort, sortColumnName, and sortDirection URL parameters to control sorting for a table, or you can create a custom sorting program. See [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#).

A JPO defined as a Sort Program must follow these rules:

- The JPO class must extend emxCommonBaseComparator JPO. The framework installs the emxCommonBaseComparator JPO. It contains the base implementations required to make custom sorting work.

```
public class ${CLASSNAME} extends  
${CLASS:emxCommonBaseComparator}  
{  
}
```

- The JPO class must define a default constructor as shown.

```
// default constructor  
public ${CLASSNAME} ()  
{  
}
```

- The class must define a method called `compare(..)`. The template for the compare method definition is:

```
public int compare(Object object1, Object object2)  
{  
    // Logic to compare the two objects and return the integer  
    value  
}
```

The input parameters and return values of the compare method are:

- param `object1`--the first object to compare.
- param `object2`--the second object to compare.
- Returns:
 - A negative integer if first argument is less than second argument.
 - A zero if the arguments are equal.
 - A positive integer if the first argument is greater than second argument.
- `object1` and `object2` is of type Map.
- The information about the sort and Map objects passed in are stored in keys field of base comparator (emxCommonBaseComparator). The keys field is a Map object containing these keys.

Key Name	Description
Type	Specifies the type of sort and should always be set to "other."
columnKey	Identifies the key name to get sort column values from Map object1 and object2.
direction	Direction of sort: ascending or descending.

Sample Sorting Code

The following sample code can be used to get column values from object1 and object2.

```
public int compare(Object object1, Object object2)
{
    // Logic to get column values to compare from object1 and
    object2
    // Get sort info keys
    Map sortKeys = getSortKeys();
    // Get column key name defined to get column values from
    object1 and object2
    String columnKey = (String) sortKeys.get("columnKey");
    // Get column values from object1 and object2
    Map m1 = (Map)object1;
    Map m2 = (Map)object2;
    String columnValue1 = (String)m1.get(columnKey);
    String columnValue2 = (String)m2.get(columnKey);
    // Code to custom compare columnValue1 and 2 goes here.
    .
    .
    .
}
```

BPS Custom Sorting Programs

Business Process Services provides two sample JPOs for custom sorting. You can use them in Engineering Central to sort Find Numbers on EBOM relationships, or as a guideline to write your own sorting program.

JPO Name	Description
emxSortNumericAlphaLarger	<p>This JPO comparator can be used to sort find numbers. The compare method works this way:</p> <ul style="list-style-type: none">• If both objects are numbers, does a numeric compare.• If both objects are alphanumeric, does a string compare.• If one object is a number and the other is alphanumeric, then Alphanumeric > Number. <p>The program first considers pure alphabetics as well as alphanumeric strings as alpha for sorting, then looks for pure numerics and sorts them.</p>
emxSortNumericAlphaSmaller	<p>This JPO comparator can be used to sort find numbers. The compare method works this way:</p> <ul style="list-style-type: none">• If both objects are numbers, does a numeric compare.• If both objects are alphanumeric, does a string compare.• If one object is a number and other alphanumeric, then Number > Alphanumeric.

Parameters and Settings for Inquiry Objects

This table describes how to fill in the parameters for inquiry objects.

For specific instructions on how to create inquiry objects using Business Modeler or MQL, refer to the [Business Modeler Guide](#) or [MQL Guide](#).

Parameter	Description	Accepted Values/Examples
Arguments	<p>The arguments replace the macros within the code. The symbolic names are translated into the actual name during substitution.</p> <p>ID is a reserved keyword for the inquiry objects used with the configurable table components. If the macro \${ID} is used in the code section, the argument ID must be assigned to any dummy value. This value is replaced by the request parameter "objectId" at run time.</p>	PART=type_Part EBOM_REL=relationship_EBOM ID = dummy
Code	<p>The code section determines the output object list for a specific business requirement. The code is generally an MQL temp query or expand bus command that selects the found objects' ids.</p> <p>At runtime, the system substitutes the macro \${ID} with the business object id. This object Id is available to emxTable.jsp, passed in from the tree node or any other commands that operate based on a specific business object. For example, typically when emxTable.jsp is called from a tree, the "objectId" parameter is passed with the specific OID assigned to the current tree node and that OID will get substituted with \${ID} at run time.</p> <p>Note that this \${ID} is not same as the RPE variable \${OID}.</p> <p>All the administrative types like Type names (Part, Buyer Desk), attribute names, relationship names must be defined as macros and each macro will be assigned with its symbolic name in the arguments list.</p> <p>All other macros (like \${PART}, \${EBOM_REL}) defined within the code are to be assigned to the appropriate symbolic names in the Argument, if required. The macros are evaluated at run time by converting the symbolic name to actual name.</p>	temp query bus \${PART} * * select id dump expand bus \${ID} from relationship \${EBOM_REL} select businessobject id select relationship id dump
Format	<p>The format determines how to reformat the line before printing the output.</p>	<p>If the table column is defined based on a list of business object IDs, the format must be a list of business object IDs separated by a new line. The format value should be \${OID} and an example output is:</p> <p>12333.3453.56765.3443 12533.3453.56765.3453 12633.3453.56765.3943 ?</p> <p>If the table column is defined based on business object and relationship IDs, the format must be a list of business object and relationship ID pairs, separated by the new line character. In each pair, the relationship ID comes first and is separated by the business object ID by a ~. The format value should be \${RELID}~\${OID} and an example output is:</p> <p>9089.34345.56567.21312~12333.3453.56765.3443 4564.3445.567.7868~12533.3453.56765.3453 6465.2342.4566.3212~12633.3453.56765.3943</p>

		?
Name	Name of the inquiry administrative object. This is the name that is referenced to evaluate this inquiry within a JSP. For naming conventions, see Naming Conventions for UI Administrative Objects .	ENCCBOM TMCPersons SCSPackages
Pattern	Indicates the expected pattern of the results of the evaluated code and shows how the output should be parsed. It sets the desired field to an Runtime Program Environment (RPE) variable or macro. The pattern is applied to each line of output from the code.	* * * \${OID} for a list of business object IDs * * * * * * \${OID} \${RELID} for a list of business object and relationship ID pairs
Test	<p>Use the Test tab to determine if the inquiry will parse the output as you have designed the JSP to expect to receive it. To test the expression entered on the Code tab, click Evaluate. The output displays on the tab. If runtime arguments are needed, enter them in the Input box. If no input is provided, the system uses the arguments entered on the Arguments tab.</p> <p>Note that you cannot use the Test tab if there are symbolic names in the code. For testing purposes, you can substitute the symbolic names in the code with actual names and then click on Evaluate.</p> <p>If the inquiry is an expand query, you must substitute \${ID} with the actual objectId (for testing only).</p>	--

Structure Browser

The structure browser component provides structure navigation capabiliites and efficient in-cell editing with a table display that supports hierarchical/structured data with large numbers of objects.

In this section:

- [About the Structure Browser](#)
- [Structure Browser View Mode](#)
- [Structure Browser Edit Mode](#)
- [Structure Browser in Flat Table Mode](#)
- [Filter for a Structure Browser Page](#)
- [Calculations for the Structure Browser](#)
- [Structure Compare](#)
- [Streaming Query and Expand](#)
- [Additional URL Parameters for Structure Browsers](#)
- [URL Parameters Accepted by emxIndentedTable.jsp](#)
- [Settings for Columns in a Structure Browser](#)
- [Parameters for Structure Browser](#)

About the Structure Browser

The structure browser component provides structure navigation capabilities and efficient in-cell editing within a table display that supports hierachal/structured data with large numbers of objects.

The following topics are discussed:

- [Introduction to the Structure Browser](#)
- [Structure Browser Main Interface](#)

Introduction to the Structure Browser

You can configure a structure browser for very specific navigation needs such as displaying the structure for EBOM/Where-Used or for general purpose navigation. You can use a relationship or JPO for expanding the structure. Users can change the view by selecting the table view from a list of tables. The expanded items are displayed in a tree structure with tabular data. The tabular data uses the administrative system tables as a configurable table component.

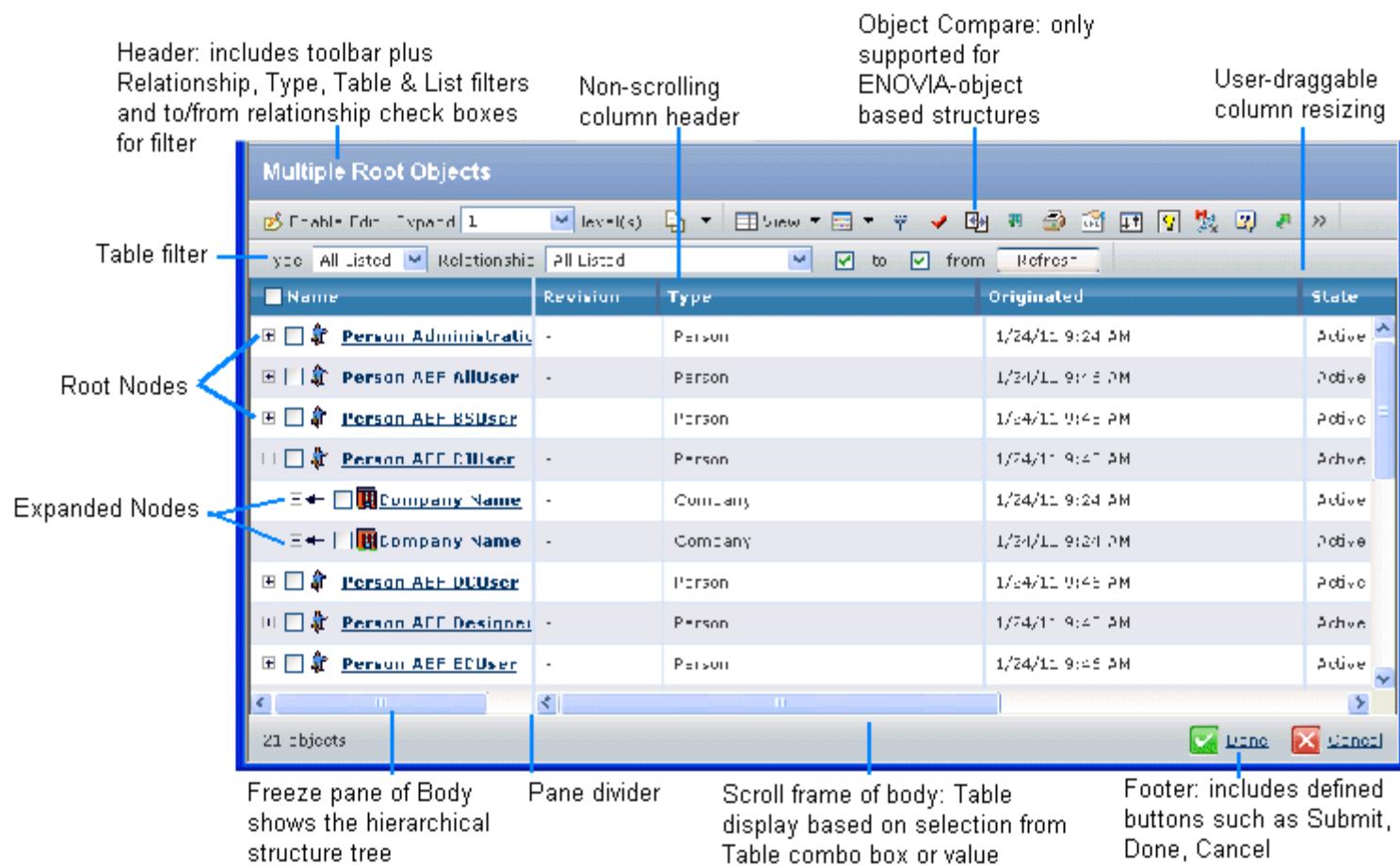
The structure browser can have a single root node, such as an EBOM, or multiple root nodes. In addition, the structure browser can display non-object based information, such as folders in an external system. A structure browser can display object-based information, or non-object based information, but not both in the same browser.

The structure browser has two modes: view (default) and edit.

The structure browser supports most of the administrative parameters supported by the configurable table component. The RowEditable key supports show or readonly, but not hide when used with emxIndentedTable.jsp

Structure Browser Main Interface

The main interface page for the structure browser component is emxIndentedTable.jsp. The following figure shows the major components of the structure browser in view mode.



For cells that contain lengthy values, the user can mouse-over the cell and a DIV pops up to display the entire contents of the cell. If the cell contains a lot of data, the DIV popup includes a scroll bar. Users can select text in the DIV, and any

hyperlinks in the original cell are active in the DIV. This functionality can be disabled by specifying the `Mouse Over Popup = disable` setting on the column. Also, if the column data is populated using a `Column Type` setting value of `programHTMLOutput` or `image`, you can define an ALT value for the column to display instead of the cell content.

In this example, the Person objects are root nodes, and the Company objects are expanded based on a relationship. If a defined column does not apply to the object for the root node, the column is left blank. For example, the Company could have a column named ZIP Code, which does not apply to a Person.

Structure Browser View Mode

View mode is the default mode for the structure browser and allows a user to look at the data and use the toolbar, but not edit cells.

In this section:

- [Structure Browser - View Mode](#)
- [Structure Browser Header](#)
- [Structure Browser Column Headings](#)
- [Structure Browser View - Table Display](#)

Structure Browser - View Mode

The structure browser displays in view mode by default. The system calls the main page emxIndentedTable.jsp with the parameter mode=view along with necessary URL parameters. If the mode parameter is not passed, the page defaults to view mode.

Example URL:

```
 ${COMMON_DIR}/  
 emxIndentedTable.jsp?objectId=14153.763.7771.42626&table=PartsR  
 eview&relationship=relationship_EBOM,relationship_DesignRespons  
 ibility&type=type_Part,type_CADDrawing&header=emxEngineeringCen  
 tral.part.eBOM&toolbar=EBOMToolbar&HelpMarker=emxebom&relations  
 hipFilter=true&typeFilter=true
```

The following figure shows the structure browser component in view mode with one tree node expanded.

The screenshot shows the Structure Browser in view mode. At the top, there's a title bar with "Invoices for: Acme Widget" and a note "(this window should be Modal, w/ Height 400px, Width 850px)". To the right is a dropdown menu labeled "Invoice Table". Below the title bar is a toolbar with various icons for Enable, Edit, Expand, and search functions. Underneath the toolbar are two dropdown menus: "Type" set to "All Listed" and "Relationship" also set to "All Listed". There are also checkboxes for "to" and "from" and a "Refresh" button. The main area consists of two parts: a tree view on the left and a table on the right. The tree view shows a single expanded node "Acme Widget" which has three children: "Invoice AW-001", "Acme Widget" (another instance), and "Order AW-001". The table on the right is titled "Invoices Table" and lists the following data:

Name	Type	Description	Invoice Numbr	Invoice Total	Invoice Date
Acme Widget	AEFQE_SBCorp	A Widget a Day Keeps the IRS Away			
Invoice AW-001	AEFQE_SBInvoi	Descr for Invoice AW-001	AW-001	1.11	2/7/11 10:39 AM
Acme Widget	AEFQE_SBCorp	A Widget a Day Keeps the IRS Away			
Order AW-001	AEFQE_SBOrde	Descr for Order AW-001			
Invoice AW-002	AEFQE_SBInvoi	Descr for Invoice AW-002	AW-002	2.22	2/7/11 4:39 PM
Invoice AW-003	AEFQE_SBInvoi	Descr for Invoice AW-003	AW-003	3.33	2/7/11 10:39 PM

At the bottom left, it says "8 objects".

Structure Browser Header

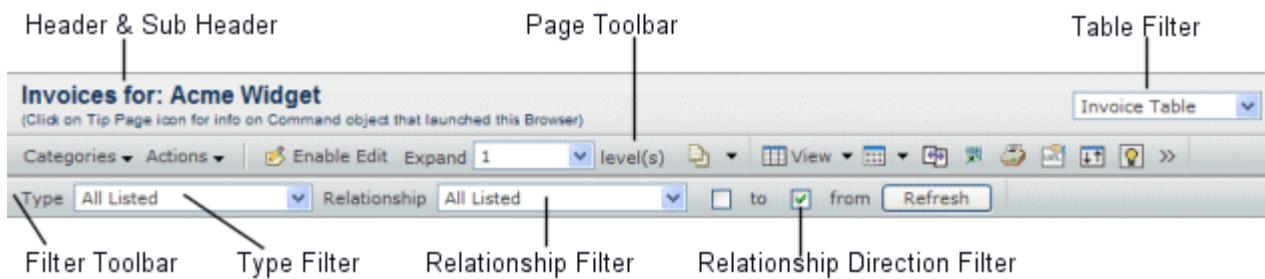
The header of the structure browser includes identifying text, toolbars, and filters.

The following topics are discussed:

- [Header Display](#)
- [Page Toolbar](#)
- [Business Process Services Supported Actions](#)
- [Filter Toolbar](#)

Header Display

The following figures shows the structure browser header section in view mode:



This table defines the header, sub header, and table filter:

Field Name	Required	Editable	Description
Header	Yes	No	Header displays using the value passed in the URL parameter <code>header</code> . If the URL parameter <code>header</code> is not passed in, the system displays the default text defined in string resource key <code>emxFramework.StructureNavigator.DefaultHeader</code> as the heading. This key is assigned to the value: Structure View: \$<type> \$<name>
Sub Header	No	No	Sub Header displays using the value passed in the URL parameter <code>subHeader</code> . It is ignored if the URL parameter <code>subHeader</code> is not passed in.
List Filter	No	Yes	The List Filter interface enables the user to change the program that expands the objects displayed in the structure browser. The system displays the option list for List Filter (in the header) only when the URL parameter <code>expandProgramMenu</code> is assigned to a menu and the menu contains more than one command. The system obtains the label displayed in the option list from the label in the command object.
Table Filter	No	Yes	The Table Filter interface enables the user to change the table view used in the structure browser. The option list for Table Filter is shown in the header only when the URL parameter <code>tableMenu</code> is assigned to a menu and the menu contains more than one command. The label displayed in the option list is obtained from the label in the command object.

The following is a URL example for the header and Table Filter display:

```
common/
emxIndentedTable.jsp?header=emxEngineeringCentral.part.eBOM&sub
Header=emxEngineeringCentral.part.AllLevels&tableMenu=ENCEBOMVi
ews
```



Page Toolbar

The page toolbar in the structure browser header is the regular toolbar shown using the URL parameter `toolbar`. The structure browser adds a toolbar button Mode (after the Actions menu), if one is defined in the toolbar menu. If the structure browser displays non-object based data, Object Compare is not supported.

The following shows the page toolbar's default display:



You can include the generic delete command, `AEFGenericDelete`, as part of a toolbar or Actions menu in a toolbar. This command provides delete functionality for the table. In Edit mode, this command immediately deletes the selected nodes (after confirmation received from user), it does not mark the rows for deletion. You can also include the `AEFGenericDeleteRMB` command as part of a right-mouse-button menu.

Action	Display	Mode	Description
Actions <User defined toolbar menu>	Actions menu	icon and text	Typically this menu is the Actions menu defined in the toolbar menu passed in as URL parameter <code>toolbar</code> .
Mode	toolbar button	icon and text	<p>You can enable the Mode button by passing the <code>editLink</code> parameter as true (default value is false). If enabled, it displays just after the toolbar Actions menu. Clicking the Mode button enables the Edit menu and displays the structure browser list in edit mode.</p> <p>The menu configured using the <code>editLink</code> parameter does not support access control mechanisms such as Role, Access Mask, and so on.</p> <p>To configure the Mode button/menu with all access control, configure the toolbar command with an HREF value assigned to "javascript:editMode()".</p>
Edit	toolbar command menu	text and icon	<p>Clicking the arrow on the button pulls-down the menu. This menu/button is defined by the AEFSBEditActions menu.</p> <p>When the structure browser changes to edit mode, the pull-down menu is enabled.</p> <p>When the Mode button is clicked again, the Edit menu is disabled.</p>
Export to Excel	toolbar	icon only	Opens the structured data as expanded in the format specified in the preference.
Printer Friendly	toolbar	icon only	Opens a printer friendly version of the current page.
Object Compare	toolbar	icon only	Opens the object compare wizard.
Help	toolbar	icon only	Opens a context-sensitive ENOVIA Help window.

In addition, column sorting is disabled if there are any pending edits (user must Apply edits before sorting).

You can also define a custom toolbar filter as defined in [Filter for a Structure Browser Page](#) that displays beneath the page toolbar.

You can define separate toolbars for Edit and View mode of the structure browser using the Mode setting on the toolbar. When the `Mode=view` setting is defined for the toolbar, that toolbar only shows in View mode. When `Mode=edit`, that toolbar only shows in Edit mode. If the Mode setting is not defined, the toolbar shows in both Edit and View mode. See [Menus and Toolbars](#) for more details.

For example, this sample URL shows 3 toolbars being passed to the structure browser:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?table=part_list  
&program=myJPO:myFunc&relationship=myrelation&toolbar=toolbar1,  
toolbar2,toolbar3&editLink=true
```

In this URL, toolbar1 could be set with `Mode=view`; toolbar2 with `Mode=edit`, and toolbar3 with no mode setting defined. In View mode, toolbar1 and toolbar3 are displayed; in Edit mode, toolbar2 and toolbar3 are displayed.

In addition, you can apply the Mode setting to submenus and commands within menus. In this case, the command is enabled or disabled depending on the current structure browser mode. For example, you can add the `Mode=edit` setting defined to a submenu for an existing menu. In View mode, the user can see but not access that submenu; in Edit mode, the user can access it.



Business Process ServicesSupported Actions

In addition to the Actions implemented specifically for the custom structure browser page, Business Process Services provides commands for Add To Selected and Remove Selected that can be included in an Actions menu. These commands are available to the user if one or more items are selected in the structure.

Add To Selected Actions Menu Command

For the Add To Selected command, you need to use a direct processing page or a custom search dialog (see [Configurable Search Webform](#)) to locate and connect the child objects.

The Add To Selected command uses the `emxTableRowId` parameter to send the Object Id, Rel Id and parent Object ID of the selected row to the custom implementation. If a custom processing page is used, it must return XML code using this XML message format:

```
<mxRoot>
<action><! [CDATA[<%= action %>]]></action>
<message><! [CDATA[<%= msg %>]]></message>
<data status="committed OR pending">
<item oid="1234.6354.3843.8237" relId="1234.6354.3843.8237"
direction="from" relType="relationship_EBOM">
    <column name="Find Number">123</column>
    <column name="Description" actual="ActualValue">DisplayValue
    </column>
</item>
<item oid="5234.7354.6843.9237" relId="4134.8354.3682.1237"
direction="from" relType="relationship_SubstitutePart">
    <column name="Find Number">234</column>
    <column name="Description" actual="ActualValue">DisplayValue
    </column>
</item>
<item oid="8664.2343.7723.9823" relId="8234.5354.7843.4237"
direction="to" relType="relationship_AlternatePart">
    <column name="Find Number">567</column>
    <column name="Description" actual="ActualValue">DisplayValue
    </column>
</item>
</data>
</mxRoot>
```

In this XML message:

- The value of the `<action>` element should be set to `add`.
- If the value of the `<message>` element contains a string, that string is shown as an alert to the user.
- The `<data status>` attribute can be "committed" or "pending". If the processing page made the connection, use committed. If the processing page did not make a connection, use pending and the connection will be made when the user clicks the Apply button.
- The direction element indicates the direction of the relationship (to or from) between parent and child objects. If not specified, the direction is not shown to the user.
- The `<item>` element contains the objectID and the relId for the added objects. If the data status is pending, it contains only the objectID. If the data status is committed, both the objectID and relId are required.
- If `<relType>` is sent, it will be processed and will be available in the markup if the user saves the edits as a markup.

If using a custom search dialog: When the user clicks Done after selecting one or more objects from the result set, the custom code should call the `editableTable.addToSelected(String XML)` function to refresh the structure browser page. If the user had selected multiple rows, the returned node(s) are added to all of the selected nodes.

When the Add To Selected functionality receives the XML message with an add action, it performs these steps:

1. Calls the database to retrieve the table column values for the returned nodes.
2. If not already expanded, expands the structure browser by one level to show the added node, unless the node would not be shown based on any filter selections in the header.
3. Rebuilds the view of the structure browser.

You can only use this option to add a child node to a selected node; you cannot use it to add a root node to the structure.

If viewing the structure browser in flat table mode, the Add To option cannot be used.

Remove Selected Actions Menu Command

You configure the Remove Selected functionality the same way as the Add To Selected feature, except the return action listed in the XML message is `remove` and there is no need for a data section.

```
<mxRoot>
<action><! [CDATA[<%= remove %>]]></action>
<message><! [CDATA[
    <%= msg %>
]]></message>
</mxRoot>
```

The Target Location setting is set to listHidden for the command to support existing HTML/JavaScript controls such as popups and messages. New implementations do not need to use the target location for processing and should encapsulate the response (message) in the mxRoot XML structure.



Filter Toolbar

The filter toolbar contains the type filter and relationship filter elements to provide a filtering interface based on

relationships, types and the relationship direction.



The following table provides the details of the elements displayed in filter toolbar.

Field Name	Required	Editable	Description
Type	No	Yes	<p>Enables users to filter the list of expanded objects based on the selected type available in the option list. The type filter control displays only when you explicitly pass the URL parameter <code>typeFilter</code> assigned to <code>true</code>.</p> <p>When the <code>typeFilter</code> parameter is passed in as true:</p> <p>The list of types passed in using the parameter <code>type</code> displays in the combo box, with an additional option "All Listed Types" selected by default.</p> <p>If you pass the <code>type</code> parameter as "all", then the combo box lists all the types in the system, along with the additional option "All", selected by default.</p>
Relationship	No	Yes	<p>The relationship filter allows the user to filter the list of expanded objects based on the selected relationship. The relationship filter control displays only when you explicitly pass the URL parameter <code>relationshipFilter</code> assigned to <code>true</code>.</p> <p>When you pass the <code>relationshipFilter</code> parameter as true:</p> <p>The list of relationships passed on the <code>relationship</code> parameter displays in the combo box, with an additional option "All Listed Relationships" selected by default.</p> <p>If you pass the <code>relationship</code> parameter as "all", the combo box displays all the relationships in the system, including an additional option "All", selected by default.</p>
To From	No	Yes	<p>The direction filter allows the user to filter the list of expanded objects based on the direction of relationship. The direction filter control displays only when you explicitly pass the URL parameter <code>directionFilter</code> assigned to <code>true</code>.</p> <p>When you pass the <code>directionFilter</code> parameter as true:</p> <p>The <code>direction</code> parameter value determines the check box "To" or "From."</p> <p>If you do not pass in the <code>direction</code> parameter or pass it as "all", the check boxes for both "From" and "To" are checked.</p>
Refresh	No	No	Refreshes the structure browser list based on the current selection in the type filter and relationship filter.

Structure Browser Column Headings

The table headings can contain text, images, or a combination of text and images.

For the structure browser, you use the same method as described in [Icons in Column Headings](#) to define an image for the heading.

The Structure Browser requires that the parameter passed to the JSP be XHTML-compliant, as described in the referenced section and in [XHTML Standard](#).

Structure Browser View - Table Display

The structure browser's table section displays below the header section. It contains a freeze pane on the left side to display the tree structure and a scrollable pane on the right to display the column data.

In this section:

- ❑ [Parts of the Table in a Structure Browser](#)
- ❑ [Root Node\(s\) Label](#)
- ❑ [Check Box](#)
- ❑ [Table Display](#)
- ❑ [Merged Cells](#)
- ❑ [Column Access Control](#)
- ❑ [Column Data Display](#)
- ❑ [Column Sort](#)
- ❑ [Grouped Columns and a Column Separator](#)
- ❑ [Custom Cell Styles](#)
- ❑ [Freeze Pane](#)
- ❑ [Expand Objects using Relationship](#)
- ❑ [Expand Objects using JPO for ENOVIA Objects](#)
- ❑ [Expand using JPO for Non-object Based Browsers](#)
- ❑ [Expand Parameters and Precedence](#)
- ❑ [Custom Expand Filter](#)
- ❑ [Example JPO Code For Custom Expand Filter](#)
- ❑ [Filters](#)
- ❑ [Manipulate Selected Rows](#)

Parts of the Table in a Structure Browser

The structure browser's table section displays below the header section. It contains a freeze pane on the left side to display the tree structure and a scrollable pane on the right to display the column data.

- The left pane contains one or more freeze pane columns, so that the user can scroll the data in the right frame and still view the freeze pane (which displays the structure tree).
- Users can resize the column width by dragging the divider between the column headers.
- The table column header stays in same position as the user scrolls the list vertically down.

The following shows the table display section:

Name	Type	Rev	Policy	F/N	Ref Des	Compon	Description
□ PT-106000-01	Part	3	EC Part				125 cc Final Assembly
□ PT-106010-01	Part	1	EC Part	001			Frame Assembly
□ PT-106020-01	Part	1	EC Part	002			125 cc Engine Assembly
□ PT-106022-01	Part	1	EC Part	003			Rear Shock, 340 psi
□ PT-106023-01	Part	1	EC Part	004			Swingarm
□ PT-106024-01	Part	1	EC Part	005			Seat, Regular
□ PT-106025-02	Part	1	EC Part	006			Gas Tank, Metal
□ PT-106030-01	Part	3	EC Part	007			Steering Assembly
□ PT-106031-01	Part	1	EC Part	001			Triple Clamp
□ PT-106132-01	Part	1	EC Part	002			O-Ring
□ PT-106033-01	Part	1	EC Part	003			Tappered Bearing
□ PT-106034-01	Part	2	EC Part	004			Handle Bar Clamp
□ PT-106040-01	Part	1	EC Part	008			Front Wheel Assembly

When opened, if the structure browser contains a single root node, it shows that node expanded to one level. If the structure browser contains multiple root nodes, only the root nodes are shown and the user can expand/collapse the nodes as needed.

Root Node(s) Label

The label for the root node is defined using the `Root Label` setting on the freezepane column.

If a root label is not defined, then `<Type> <Name> < Revision>`, such as `Part Part-5000-A 0`, is used.

You can define a string resource id, which can include a valid MQL expression, to support both internationalization, and multiple root nodes. For example:

```
emxFramework.Common.RootLabel=$<type>$<name>: Root
```

If the defined expression does not apply to the root object, such as an attribute not associated with that object, the `Root Label` value is ignored and the default label is used.

Check Box

If the structure browser is configured with a single root node and the `selection=multiple` URL parameter is also passed, then you can choose whether to show the check box for the root node.

If the structure browser is configured with a single root node and the `selection=multiple` URL parameter is also passed, then you can choose whether to show the check box for the root node. The `hideRootSelection` URL parameter controls this check box. When false (the default), the check box shows; when true (shown here) the root node does not show a check box and cannot be selected.

Invoices for: Acme Widget					
(Click on Tip Page icon for info on Command object that launched this Browser)					
Categories		Actions		Enable Edit	
Type	All Listed	Relationship	All Listed	<input type="checkbox"/> to	<input checked="" type="checkbox"/> from
Invoices Table					
Name	Type	Description	Invoice	Invoice	Inv
<input type="checkbox"/> Acme Widget	AEFQE_S	A Widget a Day Keeps the IRS Away			
<input type="checkbox"/> Invoice AW-001	AEFQE_S	testqwqe	AW-001	123.0	
<input type="checkbox"/> Invoice AW-002	AEFQE_S	Descr for Invoice AW-002 werwe werre	AW-002	2.22	
<input type="checkbox"/> Invoice AW-003	AEFQE_S	Descr for Invoice AW-003	AW-003	3.33	
<input type="checkbox"/> Invoice AW-004	AEFQE_S	Descr for Invoice AW-004	AW-004	4.44	

If the `selection` URL parameter is set to `single` or `none` or the structure browser is configured with multiple root nodes, then the `hideRootSelection` URL parameter is ignored.

Table Display

The table view displays based on a system administrative table configured in the Business Modeler.

You pass the table name to the structure browser component emxIndentedTable.jsp via the URL parameter `table`. To pass more than one table view with the table filter interface pass the `tableMenu` URL parameter with the administrative menu name.

To enable the table filter, pass the URL parameter `tableMenu` with a value of the administrative menu name. You must assign a label to each command to display as the label in the table filter pick list. For internationalization, the label can be a string resource key but it must be associated with the setting `Registered Suite`. The commands connected to the menu honor all the Access settings supported by the configurable component (like Access Mask, Access Expression, and Access Program).

If there is only one command, the system uses the command setting to display the table view, but there is no table filter.

To change the table view, change the table option listed in the table filter.

You can also assign the `tableMenu` parameter to an administrative command. If this is a command name, the system uses the command setting `table` to get the table definition and the table filter does not display.

Example URL using `table` and `tableMenu`:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?objectId=14153.763.7771.42626  
&table=PartsReview&relationship=relationship_EBOM,relationship_DesignResponsibility  
&type=type_Part,type_CADDrawing&header=emxEngineeringCentral.part.eBOM&toolbar=EBOMToolbar.  
 ${COMMON_DIR}/emxIndentedTable.jsp?objectId=14153.763.7771.42626  
&tableMenu=ENCEBOMView&relationship=relationship_EBOM,relationship_DesignResponsibility  
&type=type_Part,type_CADDrawing&header=emxEngineeringCentral.part.eBOM&toolbar=EBOMToolbar
```

If you pass both `table` and `tableMenu` as part of the URL, the `table` parameter takes precedence and `tableMenu` is ignored.

The structure browser supports the URL parameter `freezePane`. This is assigned to a column name that displays as the freeze pane column. If you do not pass this parameter, the system uses the first column in the table as the freeze pane column. All other columns are displayed in the right pane.

When the user changes the table view by selecting a different table from the table filter, the table display changes but the freeze pane column remains the same. However, if the new table does not contain a column matching the current freeze pane column, the first column of the new table displays in the freeze pane column.

When the user expands nodes (by clicking on the plus sign), the nodes expand to one level at a time and the corresponding rows on the right are synchronized with those on the left. For example, a Bill of Materials could have a full assembly and sub-assembly tree which remain in view while scrolling a large number of columns on the right.

You may need to modify any customized actions that you have added to the Menu ENCBOActionsToolBar to handle the expanded table. For example, can a customized action can be performed on released parts?

Merged Cells

You can create merged cells in the structure browser.

You can create merged cells in the structure browser using these settings:

- Group Name
- Row Span
- Row Number

The Group Name setting on a column indicates it will be merged with all other columns that have the same Group Name value. The Row Span setting indicates how many rows that column will span and can be set to 1 or 2. Row Number determines in which row in the group that column's data will be shown.

The width of a Group is determined by the width of the column specified for Row 1 of the group. In all cases, if the user does not have access to a column, then that column is not displayed.

For example, if the Name and Type columns are grouped together, and the Description column has the Row Span = 2 value set, the table looks like:

Name	Description
Type	
123-456	This is the description of the object for this row and could be fairly long, so spanning 2 rows makes the table easier to read.
Part	
123	This is the description of the object for this row and could be fairly long, so spanning 2 rows makes the table easier to read.
Part Assembly	

In the above example, the Name column would be defined with Row Number = 1, and the Type column would be defined as Row Number = 2. If both the Name and Type columns have a Width setting defined, then the width for Row 1 (Name) is used.

You could also include all 3 columns in the same group. The result would look like this:

Name	Type
Description	
123-456	Part
This is a long description that spans the Name and Type columns.	
123	Part Assembly
This is a long description that spans the Name and Type columns.	

For this layout, the columns would have these settings:

- Name
 - Group Name=Gr3
 - Row Number=1
 - Row Span=1
- Type
 - Group Name=Gr3
 - Row Number=1
 - Row Span=1
- Description
 - Group Name=Gr3
 - Row Number=2
 - Row Span=1

If these columns have Width settings, then the total width for the Name and Type columns is used, and the Width value for Description is ignored.

When a structure browser includes groups of 2 rows, all columns not in the group automatically span the 2 rows. You can explicitly define the Row Span setting for those columns, or accept the default.

Column Access Control

The structure browser component uses the same access control mechanism used in other configurable components.

The structure browser supports the following types of configurable access for a given column:

- Role Access
- Access Mask
- Access Expression
- Access JPO.

For the complete details on how the access control is configured, see [User Access to UI Components](#).

Column Data Display

The rows in the table cannot wrap because the synchronization between the left and right panes would be lost. Instead the text is displayed in a mouseover of the cell as Alt text.

As an alternative, you can define merged cells to provide more room for text. See [Merged Cells](#) for instructions. If a subset of the merged cells are also defined to display in the freeze pane, then those columns are removed from the merged group (and display appropriately in the freeze pane), and the remaining cells are adjusted accordingly.

In general all the column level capabilities supported by the standard configurable table are available in the structure browser component. For example, you can configure every column to show the type icon, hyperlinked, and so on.

For the root objects in the structure, the column data displays only for those columns where applicable.

Column Sort

You can define the initial sort order, and which columns are allowed to be used for sorting in the structure browser.

Users can sort the columns by clicking on the header text link or the  toolbar button.  opens a dialog box allowing the user to choose up to 3 columns for sorting. Only columns defined with the Sortable=true setting can be selected. In addition, columns with the sortType=other setting defined cannot be used for multiple column sorting. See the *Application Exchange Framework User's Guide* for details.

When the page displays the first time, the initial sorting is done based on the URL parameter `sortColumnName` and `sortDirection`. If you do not pass this URL parameter, the first column (the freeze pane column) is used for sorting the structure. You can pass up to 3 column names with the `sortColumnName` parameter and 3 directions with the `sortDirection` parameter. In addition, you can pass `sortColumnName=none` so that no initial sorting is done.

When the user opens the Sort by dialog box (by clicking  in the page toolbar), the 1st, 2nd, and 3rd column name fields are populated based on the values passed to the `sortColumnName` and `sortDirection` URL parameters. If less than 3 values are passed, only the values passed show populated in the dialog box and the remaining fields are blank.

If the user creates a custom table based on the system table for the structure browser page, the sorting options selected by the user determine the default values for the Sort by dialog box.

If the `sortColumnName` is used for sorting, the subsequent node expansion uses the specific sort column name to sort the child objects for that page.

If the user clicks on one of the other column headers to sort, the system updates the page level `sortColumnName` setting to the new column. Subsequent expands will use this new `sortColumnName` to sort the objects. If the user clicks on any column header for sorting while the structure is already expanded to multiple levels, the sorting is done within the individual levels and redisplays the sorted structure with all the nodes as expanded before sorting.

For pages that include columns with the type `program` or `programHTMLOutput` with complex logic, the sort process may take a while. You can set the Sortable setting on the column to false if needed.

Grouped Columns and a Column Separator

You can configure column groupings in the structure browser by adding a header over several consecutive columns and separating the columns with a separator.

To configure this group header, use the `Group Header` setting for each column in the group. To add a separator, use the `Column Type=separator` setting. If some of the columns in the group are defined to display in the freeze pane, then the group header shows over those columns, and is repeated over the remaining grouped columns in the scroll pane. If you define the columns to display on either side of the pane separator, they will retain the look of a group (although the heading is still repeated on both sides of the pane separator).

Custom Cell Styles

You can define custom styles for cells in the structure browser table.

You can use these column settings to define customized styles for cells in a column:

- `Style Column`
- `Style Program`
- `Style Function`

The `Style Column` setting provides the name of a definition in the dsecUIType-Custom.css (a program administrative object) to use for the cells in that column. For example, you could add this setting to the column:

```
Style Column = ColumnBackGroundColor
```

The dsecUIType-Custom.css file would include this definition:

```
ColumnBackGroundColor {  
    background-color : rgb(252, 186, 186);  
}
```

All cells in that column would show with that background color.

You can also use the Style Program (specifies a JPO) and Style Function (specifies the method within the JPO) settings to determine the style used to display individual cells in the column. The JPO:method accepts an object list and returns a stringlist of CSS specification styles for each object in the object list. Each cell could have a different style specification, and some cells could have no style defined.

You can use the expand program (see [Expand Objects using JPO for ENOVIA Objects](#)) to include a styleRows value for each object map of the returned object list. The value for styleRows must be a CSS specification style.

If a structure browser includes multiple methods for setting cell style, such as using both the Style Column setting on the column and a styleRows value in the object list, the structure browser uses this order of precedence:

- `Style Program` and `Style Function` settings on the table column
- `styleRows` setting in the returned object list
- `Style Column` setting on the table column

In the above example, the styleRows cell style overrides the Style Column cell style.

Freeze Pane

The structure tree displays in the freeze pane on the left side of the table display. To display the structure tree, the structure browser component must have one or more a root objects.

The structure tree displays in the freeze pane on the left side of the table display. To display the structure tree, the structure browser component must have one or more a root objects.

For a single root node, you define the root object by passing the URL parameter `objectId` (a required parameter for this component if configured for a single root node).

For multiple root nodes, you pass the URL parameter `program`, where `program` is the name of JPO and method that returns a MapList. Each map in the list has a key `id` assigned to an Object Id for the corresponding root object. The Root Label setting on the freezepane column is applied to all root objects retrieved in the MapList.

With a given `objectId` value or MapList, you can obtain the list of objects to be displayed in the next level of the structure tree by one of the two interfaces supported by the structure browser component.

- Define one or more `relationship` names with an optional `direction` parameter (passed in as URL parameters).
- Define a JPO to return the list of objects to display as the child objects for a given `objectId`. To use the JPO to expand, pass the URL parameter `expandProgram`.

If the structure browser is non-object based, the `relationship/direction` parameters cannot be used.

The details are outlined in the following sections.

Expand Objects using Relationship

You can use a relationship to expand the object under the root node when you have a object root nodes; you cannot use the relationship to expand a structure browser that contains non-object based root nodes.

You pass one or more relationship names to the structure browser component as the URL parameter `relationship`. Pass the relationship name as the symbolic name or the actual name. To pass multiple relationships, use a comma-separated list. Pass the `direction` parameter with the value `from` or `to` to filter the relationship objects based on the direction.

For example, for a single root node:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?objectId=14153.763.7771.42626  
&table=PartsReview,PartsReleased&relationship=relationship_EBOM,relationship_DesignResponsibility  
&direction=from&type=type_Part,type_CADDrawing&header=emxEngineeringCentral.part.eBOM&toolbar=EBOMToolbar
```

For multiple root nodes:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?table=PartsReview&program=  
ECPart;getMyPartList&relationship=relationship_EBOM,relationship_DesignResponsibility  
&type=type_Part,type_CADDrawing  
&header=emxEngineeringCentral.part.eBOM&toolbar=EBOMToolbar&  
HelpMarker=emxhelpebom&relationshipFilter=true&typeFilter=true
```

The root object is expanded for the given relationship(s) and the direction. If you pass more than one relationship name, the object is expanded using all the relationships. The direction parameter is optional. By default both directions are expanded. If the expand is either `from` or `to`, you must pass it explicitly with the direction parameter. When the structure tree is expanded, the line connecting the parent and child node indicates the direction, with an arrow pointing towards the right direction.

Expand Objects using JPO for ENOVIA Objects

You can use a JPO to control the expansion of a structure browser.

Pass the JPO name with the function name to the structure browser component as the URL parameter `expandProgram`. Assign the parameter value as the JPO name and the JPO method name, separated by a colon, as `<JPO Name>:<JPO method Name>`.

If you plan to use more than one expansion JPO with the List Filter interface, pass the URL parameter `expandProgramMenu` with a value of the administrative menu name.

To enable the List Filter, pass the URL parameter `expandProgramMenu` with a value of administrative menu name. The menu should contain more than one command, with each command setting `program` and `function` assigned to the JPO name and the method name. Assign each command to a label, that is displayed as the label in the List Filter pick list. For internationalization, the label can be a string resource key. It must be associated with the `Registered Suite` setting. The commands connected to the menu honor all the access settings supported by the configurable component (such as Access Mask, Access Expression, and Access Program).

If only one command is connected, the system uses the command setting to get the expansion JPO, and it does not display the List Filter.

You can assign the `expandProgramMenu` parameter to an administrative command. If it is a command name, the system uses the command settings `program` and `function` to get the JPO and method name and the List Filter is not shown in the header.

The system provides the JPO with all the necessary input parameters including the `objectId`. The input and the output from this JPO will be similar to the approach used in standard configurable table to get the object list. The output is a `MapList` object with a list of `HashMap`s. Each `HashMap` object has the keys `id` and `id[connection]` with the values of the child object's id and the connecting relationship's id.

You can also pass the URL parameter `direction` with the value assigned `from` or `to`, to filter the objects based on the direction. However, you must be sure to filter the list with the `expandProgram` approach. Follow one of these strategies:

- Write the JPO to return the `MapList` with `HashMap` objects. Each `HashMap` must contain an additional key `direction` assigned to `from` or `to` or `both`.
- Write the JPO to filter the objects based on the request parameter `direction` passed in. Return the filtered object list.

For example, for a single root node:

```
 ${COMMON_DIR}/  
 emxIndentedTable.jsp?objectId=14153.763.7771.42626&table=PartsReview,PartsReleased&tableLabel=Review,Released&expandProgram=emxPart:getEBOMs&direction=from&type=type_Part,type_CADDrawing&header=emxEngineeringCentral.part.eBOM&toolbar=EBOMToolbar
```

```
 ${COMMON_DIR}/  
 emxIndentedTable.jsp?objectId=14153.763.7771.42626&table=PartsReview,PartsReleased&tableLabel=Review,Released&expandProgramMenu=ENCBOMList&direction=from&type=type_Part,type_CADDrawing&header=emxEngineeringCentral.part.eBOM&toolbar=EBOMToolbar
```

For multiple root nodes:

```
 ${COMMON_DIR}/  
 emxIndentedTable.jsp?table=PartsReview&program=ECPart:getMyPartList&expandProgram=ECPart.getEBOMs&header=emxEngineeringCentral.part.eBOM&toolbar=EBOMToolbar
```

Expand using JPO for Non-object Based Browsers

When using a structure browser to display non-ENOVIA objects, the root node may or may not be associated with an ENOVIA object, however, all of the connected nodes must be non-object based. You cannot mix ENOVIA objects and non-ENOVIA objects in the same structure browser list.

The JPO method for the expandProgram returns a MapList. Each map represents a node and contains the node-specific information. This table lists the parameters supported:

Key Name	Description	Possible / Default Values
id	Required key information used to identify the node in the structure. It must be assigned to any unique id.	<object ID> for an ENOVIA object <unique ID> for external data
id[connection]	Relationship id Not applicable for non-object based data.	<rel id> for an ENOVIA object
direction	to / from not applicable for non-object based data.	to from
selection	Controls whether or not to include a column of check boxes or radio buttons as the first column.	single (shows radio buttons) multiple (shows check boxes) none (first column contains data)
RowEditable	Controls whether or not that row is editable.	show readonly
disableSelection	Controls whether check boxes on the structure browser page are enabled or disabled.	true (default) false

Expand Parameters and Precedence

3 URL parameters can be used to expand a structure browser. You need to understand the order of precedence followed to expand a structure browser.

The following three URL parameters are the main parameters that control the expansion of objects in the structure navigation. These parameters are listed in the order of precedence honored by emxIndentedTable.jsp. If more than one of these parameters are passed in together in the URL, the following precedence applies and the other is ignored:

- `expandProgram`
- `expandProgramMenu`
- `relationship`

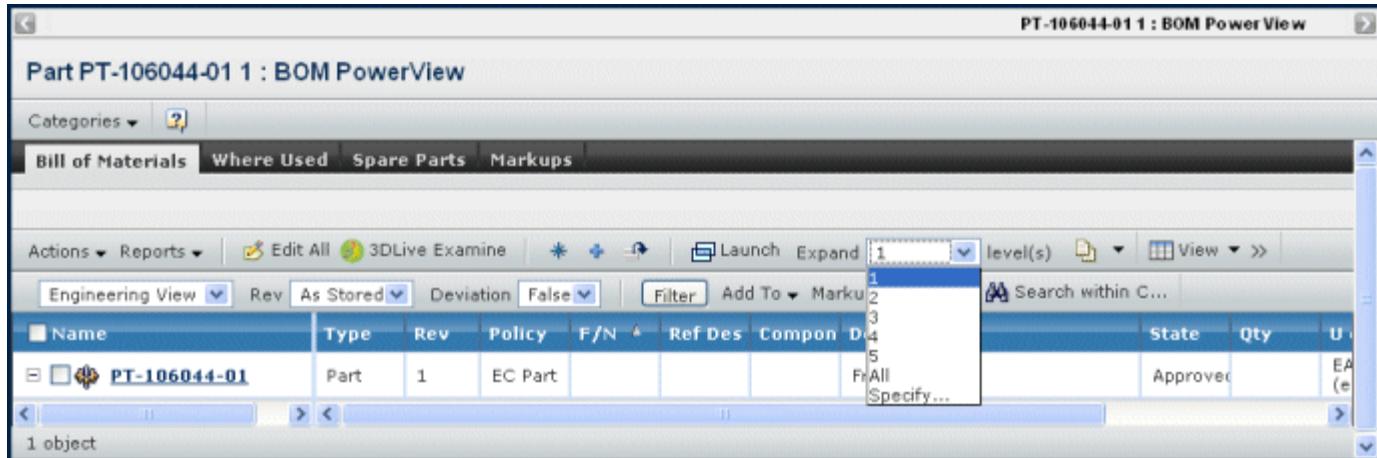
You cannot use the `relationship` parameter to expand a non-object based structure browser; you must use either `expandProgram` or `expandProgramMenu`.

One of the above parameters is a required parameter. If none of the parameters is passed in, the `relationship` parameter is used with the default value "all".

Custom Expand Filter

You can add a custom expand filter to the structure browser's toolbar

This example shows a custom filter:



The framework includes the `AEFFreezePaneExpandLevelFilter` menu that can be used for any structure browser page, or you can write a custom menu. This menu contains these commands:

- `AEFFreezePaneExpandNLevel` (provides range values 1 through 5)
- `AEFFreezePaneExpandAllLevel`
- `AEFFreezePaneExpandMoreLevel` (corresponds to the `Specify` value)

You can use these administrative parameters on the menu command:

Parameter	Description	Accepted Input Value
Access	The user, group, or role to which this option applies.	All Test Everything
Label	The display values for choices in the Expand combo box. A string resource property can be provided. If a label has not been defined, then the Level information shows.	Next,2,3,4,5 emxFramework.FreezePane.Next

The expand filter is controlled by these URL parameters:

- `expandLevelFilter`--enables or disables the expand filter
- `expandLevelFilterMenu`--defines the administrative menu that adds the expand filter to the toolbar

The `Level` setting defines the values that populate the Expand combo box. The `expandProgram` JPO (see Expanding Objects using JPO for Matrix Objects) uses the value selected in this box when expanding a node. The values for the `Level` setting can include the keywords All and Specify:

- `All`--indicates that nodes will be expanded to the leaf level.
- `Specify`--opens a text input box where the user enters a specific value (must be a positive integer).

Both All and Specify are supported by the relationship expand and an `expandProgram` JPO. The Labels for the All and Specify options are displayed based on the Label parameter for the command object, and can be internationalized.

You can include a custom text value as a level if you write a custom `expandProgram` JPO that processes that value. You cannot use a custom text value with a relationship expand.

When a structure browser page is loaded, the Expand filter shows the first value entered for the first command in the expand menu. If that command is a comma-separated list, the first value in the list is used. If the structure browser shows multiple root nodes, the page is not expanded to any level when initially loaded.

When a user clicks + for a node, the shown value is used for the expand process. If the user changes the value, already expanded nodes are not updated: the user must collapse and expand that node for the new expand value to take affect.

If the type or relationship/direction filters are used, the Refresh button causes the structure to expand one level at a time to the level selected in the Expand levels combo box using the selected filter options. The user cannot select both the To and From directions if the Level is set to All--this combination would create an infinite loop. If the user tries this combination, an

error message displays.

If you anticipate that the expand could return large object sets, you should set a value for the BOS_EXPAND_LIMIT setting in the enovia.ini file for the ENOVIA Live Collaboration Server to limit the number of objects returned from the ENOVIA Live Collaboration core. You can also use the Studio Customization Toolkit to set a limit to the expand (the Studio Customization Toolkit expand limit can be used within the expandProgram JPO).

Example JPO Code For Custom Expand Filter

The JPO should expand the structure 1 level at a time, even when a larger value (such as 2 or 3) is selected as the expand level. If you define a custom expand text value, the JPO is responsible for filtering the objects returned based on that value.

This code shows the structure of an expandProgram JPO:

```
public MapList expandJPOName (Context context, String[] args)
throws Exception
{
    // Get input HashMap
    //requestMap-contains all the request parameters, including the
    key parameters like objectId, relId, languageStr and custom
    filter, expand filter field values, default level to expand,
    HashMap requestMap = (HashMap)JPO.unpackArgs(args);
    // Get object id
    String sObjectId = paramMap.get("ObjectId");
    // Get expand level
    String sLevel = requestMap.get("Expand Level");
    // Do necessary logic to process the level info and then return
    the expanded objects
    /*
    **
    ** The code to use object id & level to get expanded objects
    **
    */
    // Return modified MapList
    return mapList ;
}
```

Filters

When the structure browser component opens for the first time, the Filter Bar in the header populates and sets the default values based on the URL input parameters: relationship, direction, type, typeFilter, relationshipFilter, directionFilter, and so on.

When the filters are initialized with a list of items, the user can change these values and press the Refresh button in the filter toolbar to display the structure based on the modified filter.

Suppose the user is viewing the structure view by expanding the nodes. The user changes the filter option, but does not press the Refresh button. If the user tries to expand another unexpanded node, the system will begin using the new filter setting for the subsequent expansion. If the user wants filtering to be applied to the complete view, they might need to click Refresh.

Manipulate Selected Rows

You can set up the structure tree to display with check boxes or radio buttons in the freeze pane column so that the user can select and process items using the custom pages.

For example, you can write custom pages to disconnect or delete the selected objects. Initially, the system displays check box or radio buttons using the URL parameter `selection` assigned to `multiple` or `single`. You can configure the toolbar command to interact with the selected items. The necessary settings are outlined below.

The `href` administrative parameter is associated with the toolbar command item configured for processing the selected rows. This parameter is assigned to a JSP file that processes the selected rows. See the following table for the settings associated with the toolbar command item configured for processing the selected rows.

Setting Name	Description
Target Location	Assigned one of the following: <code>listHidden</code> --if the processing is purely a background action and no UI is required. <code>popup</code> --if the processing requires a user interface dialog. (If the value is <code>popup</code> , use the associated settings such as "Popup Modal").
Submit	Assigned to <code>true</code> so that the selected items in the structure are posted to the processing page.
Confirm Message	Configures a custom message.
Row Select	Assign to <code>single</code> or <code>multi</code> to control the selection.

The processing page gets the selected items as the posted data. The following request parameters are available to the processing page:

- `objectId`--Contains one or more object IDs of the root objects in the structure tree.
- `emxTableRowId`--Gives all the selected items as a string or array of strings. The values can be read using servlet APIs.
- `request.getParameter()`--returns string
- `request.getParameterValues()`--returns string array

Each value for `emxTableRowId` contains: `relId`, `objectId`, and `parented` for the selected node, separated by pipe "|", in the format `<relId>/<objectId>/parentId>`.

1. `relId`--relationship ID for the selected node
2. `objectId`--object ID of the selected node
3. `parentId`--the object ID of the parent node to be the selected node

The processing page uses the passed-in values and processes them as required. Sends a message back to the structure browser component as the desired action when processing completes.

The processing JSP must return the following XML:

```
<mxRoot>
<action><! [CDATA[<%= action %>]]></action>
<message><! [CDATA[<%= msg %>]]></message>
</mxRoot>
```

In the above XML, `action` is a JSP variable. It can also be a static string. Its value should be one of the following:

- `remove`--Removes the selected nodes from the display
- `refresh`--Refreshes the structure view
- `error`--Error condition
- `void`--No action

The XML also contains the variable `msg`. It can be assigned any text to display as an alert message.

The following shows a JSP sample/template for the processing JSP:

```
<%@include file = ".../common/emxNavigatorInclude.inc"%>
<%
String action = "remove";
String msg = "";
//read the necessary parameters from the posted data
String tableRowIdList[] = emxGetParameterValues(request,
"emxTableRowId");
String objectId = emxGetParameter(request, "emxTableRowId");
try {
    ContextUtil.startTransaction(context, true);
```

```
if (tableRowIdList!= null) {
    for (int i=0; i< tableRowIdList.length; i++) {
        System.out.println("tableRowId :::: " +
tableRowIdList[i]);
        //process - relId|objectId|parentId - using the
tableRowId
        String tableRowId = tableRowIdList[i];
    }
}
} catch (Exception ex) {
    ContextUtil.abortTransaction(context);
    action = "error";
    if (ex.toString() != null && (ex.toString().trim()).length()
> 0){
        msg = ex.toString().trim();
    }
} finally {
    ContextUtil.commitTransaction(context);
}
//clear the output buffer
out.clear(); %>
<mxRoot>
<action><! [ CDATA[<%= action %>]]></action>
<message><! [ CDATA[ <%= msg %>]]></message>
</mxRoot>
```

Structure Browser Edit Mode

The structure browser component supports table data editing.

In this section:

- [About Edit Mode](#)
- [Custom Logic for Connections in the Structure Browser](#)
- [Validating Edit Operations](#)
- [Sample JPO Program for connectionProgram](#)
- [Pre/Post/Cancel Processing for Editable Structure Browser](#)
- [Mass Update Toolbar](#)
- [In-cell Editing](#)
- [Markup Interface](#)
- [Editable Columns](#)
- [Adding Rows for New or Existing Objects to the Structure](#)
- [Validating Cell Values](#)
- [Custom Client-side Validation](#)
- [JavaScript APIs Available for Validation Methods](#)
- [XHTML Standard](#)

About Edit Mode

The structure browser component supports table data editing.

You can configure the Mode toolbar button and Edit menu to show up in the view mode in one of these ways:

- By passing the `editLink=true` parameter to `emxIndentedTable.jsp`--This approach adds the Mode button and Edit menu in the toolbar. This button displays for all users who have access to the structure view page--you cannot control user access. By default, the `editLink` parameter is false. The Edit menu is configured by connecting the `AEFSBEditActions` command to the toolbar.
- By configuring a command to invoke the structure browser in edit mode--The command must be connected to the toolbar menu, which is passed in as the `toolbar` parameter. You must assign the command href parameter to `javascript:editMode()`.

The user can toggle between the view and edit mode by clicking on the Mode button. In view mode, the Edit pull-down menu is disabled. When the user clicks the Mode button, the following changes happen in the structure browser:

- The Filter Toolbar changes to the Mass Update Toolbar
- The table displays within the structure view. Visual color changes indicate those cells and headers of columns that are editable.
- The Apply and Reset buttons display in the toolbar.

If the Cancel or Submit buttons display in the footer, they are grayed out so the user will not confuse their usage with the Apply/Reset buttons in the toolbar.

- The Edit pull-down menu is enabled. See the *Application Exchange Framework User's Guide* for details on using the menu items.

If the initial data load for the structure browser used the `program` parameter, the `RowEditable` setting has a default value of `true` (typical for structure browsers with multiple root nodes). If the data load is based on an `objectId` (using the `getRelatedObjects` method in the JPO defined by the `expandProgram` parameter), then the `RowEditable` setting has a default value of `false`.

When the user cuts or copies a row, the system stores the relationship and direction used by that connection. When the cut or copied row is pasted, the system uses the relationship/direction to make the connection to the new parent object. If a relationship URL parameter was passed to the structure browser, the system selects a valid (for the combination of child/parent object) relationship to make the connection. To implement a different connection method, see [Custom Logic for Connections in the Structure Browser](#) for details.

You can control which child rows can be cut and pasted using the `editRelationship` URL parameter. When a value is passed for this parameter to the structure browser, only child objects connected to the parent objects using one of the specified relationship can be cut and pasted. If this parameter is not passed, there are no restrictions on which child objects can be cut and pasted. The `connectionProgram` and `expandProgram` URL parameters also determine which objects can be edited, based on their internal logic.

In addition, you can control which child rows can be cut/pasted from/to the same parent object. For example, if the sequence of child objects is significant to the business process, the user may want to move a child from one location in the list to another, while retaining the same parent object. The `resequenceRelationship` URL parameter defines which relationships permit this action. For example, if this parameter is passed as `resequenceRelationship=relationship_EBOM`, then only child objects connected with this relationship can be cut/pasted from/to the same parent object. If connected with a different relationship, it can still be cut, but must be pasted to a different parent object (depending on the value of the `editRelationship` URL parameter).

When the user clicks on the Mode button again, the page toggles back to view mode. The following figure shows the structure browser in edit mode:

Mass Update Toolbar	Enables/Disables Editing	Apply button allows user to make incremental changes without submitting entire page	Editable Field Indicator
---------------------	--------------------------	---	--------------------------

Name	Type	Description	Invoice N	Invoice Total	Tr
Acme Widget	AEPQE_SE	A Widget a Day Keeps the IRS Away			
Invoice AW-001	AFTQF_BP	Description for Invoice AW-001	AW-001	1.11	
Invoice AW-002	AEPQE_SE	Description for Invoice AW-002	AW-002	2.22	
Invoice AW-003	AFTQF_BP	Description for Invoice AW-003	AW-003	3.33	
Invoice AW-004	AEPQE_SE	Description for Invoice AW-004	AW-004	4.44	

In edit mode, there is a visual cue to indicate that some of the columns / cells in the table are editable. There are visual cues to indicate the edits after the user has clicked outside the editing area. The system displays an Apply button in the edit mode of the table so that the user can apply changes. This eliminates the need to submit a massive number of forms in a single submit. It also eliminates the need to draw a table with thousands of form elements.

⊖--Row is marked for deletion.

▲--One or more values in the row were changed, shown as 20 10.

✚--New row added.

↳--Row was moved (cut and pasted) from another location in the structure for the same parent.

Child rows of newly-added rows can be edited prior to applying the edit that added the row.

Custom Logic for Connections in the Structure Browser

When editing in a structure browser, the system automatically uses the relationship and direction used by the copied/cut object or passed in using the `relationship` URL parameter. That relationship may not be valid for the new parent object, or the user may not have connect access. Either of these situations displays an error to the user.

To implement a different method of connecting children to parent objects, you need to write a custom JPO and pass its name to the structure browser using the `connectionProgram` URL parameter.

The method signature is:

```
public static Map connectDocument(Context context, String[] args) throws Exception
```

The input `args[]` array contains a packed map with the entries defined in this table.

Key	Content
paramMap	A map containing key/value pairs of the request parameters originally passed to <code>emxIndentedTable.jsp</code> .
parentOID	The object ID of the parent side of the connection.
objectId	The object ID of the child side of the connection.
relId	The relationship ID used by the original connection from where the child object was cut or copied.
level	Level information.
sequence	The sequence of child objects in a given level.
action	One of these values: remove add resequence none
contextData	The JDOM XML structure.

The method output is a Map that should look like this:

Key	Value
Action	SUCCESS--update completed successfully ERROR--updated failed
Message	Either a string resource key or static text explaining any errors that occurred. The message will be displayed to the user.
changedRows	MapList of all child HashMaps. Each HashMap contains the key value pairs for <code>rowId</code> , <code>relid</code> , <code>oid</code> , <code>markup</code> .

If the Map includes a value for the `Message` key, the value for the `Action` key is checked. If `Action=SUCCESS`, the changes within the transaction are all committed to the database. If `Action=ERROR`, the entire transaction (all changes marked up within the structure browser) are aborted and not committed to the database. If the JPO does not need to return anything to the structure browser, it can return nothing or an empty map and the `Action` is assumed to be `SUCCESS`.

Validating Edit Operations

If you provide a custom connectionProgram, you must validate the edit operations prior to committing them to the database.

The structure browser can be passed the relationship or the `expandProgram` parameter that defines what children to show for a parent object. Whether edit actions are permitted depends on the URL parameters passed to the structure browser:

- If using the `relationship` (and no `expandProgram`) parameter, edit operations (cut and paste) are only allowed if the URL also includes the `editRelationship` parameter AND the child object is connected to the parent object using one of the relationships defined by the `editRelationship` parameter.
- If using the `expandProgram` parameter, edit operations are only allowed if the returned HashMap for the specific row contains the key `allowEdit=true`.
- If the `expandProgram` and `editRelationship` parameters are passed, then the row must have `allowEdit=true` AND the relationship that connects it to the parent object must be one of the relationships defined by `editRelationship`.

To use resequencing functionality, the structure browser must meet one of these conditions:

- For an object-based structure browser: the `resequenceRelationship` URL parameter in addition to the `connectionProgram` parameter must be passed.
- For a non-object-based structure browser: The `expandProgram` and `connectionProgram` parameters must be passed and the returned HashMap for the specific row must contain the key `allowEdit=true`.

Sample JPO Program for connectionProgram

This sample shows a JPO that can be used to connect objects in the editable structure browser.

```
public static Map connectPartToDocument (Context context,
String[] args) throws Exception
{
    // unpack args array to get input map
    MapList programMap = (HashMap) JPO.unpackArgs(args);
    // get request information
    MapList paramMap = (HashMap) programMap.get("paramMap");
    // get parentobjectId
    String parentId= (String) programMap.get("parentOID");
    // get childobjectId
    String childId= (String) programMap.get("objectId");
    // create a relationmap to be returned.
    Map connectionMap = new HashMap();
    //Create MapList that holds all the changedRows (added/removed/
    resequenced objects i.e each add/removed/resequenced object is
    a HashMap having all details oid,rowId,relid and markup key
    values)
    MapList mlchangedRows=new MapList();
    // format for MapList mlchangedRows
    {
        {rowId=0,oid=42864.18383.13045.21601,
        relid=52728.49964.37568.33109, pid=52728.49964.13046.12899,
        markup=add},
        {rowId=0,oid=52728.49964.13046.28148,
        relid=52728.49964.37568.34361, pid=52728.49964.13046.12899,
        markup=add},
        {rowId=0,oid=57660.21992.15800.53540,
        relid=52728.49964.37568.36206, markup=cut}
    }
    // Custom logic to create the connection between parent object
    and child object goes                                //here..
    //Populate the return map
    connectionMap.put("Action", "SUCCESS" );
    connectionMap.put("Message", "errormessage" );
    connectionMap.put("changedRows", mlChildrenObjects);
    return connectionMap;
}
```

Pre/Post/Cancel Processing for Editable Structure Browser

The system invokes preprocessing before loading the editable structure browser page. It invokes cancel processing when a user clicks a Cancel or View button, or closes the editable structure browser component.

In this section:

- [About Processing the Structure Browse Edit View](#)
- [Pre Process URL for an Editable Structure Browser](#)
- [Pre Process JPO for an Editable Structure Browser](#)
- [Cancel Process URL for an Editable Structure Browser](#)
- [Cancel Process JPO for an Editable Structure Browser](#)
- [Post Process URL for an Editable Structure Browser](#)
- [Post Process JPO for an Editable Structure Browser](#)

About Processing the Structure Browse Edit View

The system invokes preprocessing before loading the editable structure browser page. It invokes cancel processing when a user clicks a Cancel or View button, or closes the editable structure browser component.

The system invokes post processing after completion of the edit process. The system performs the edit process, post process JSP, and post process JPO in a single transaction. If any exception happens in the edit process or post process, the system rolls back the entire transaction.

You can specify a pre/post/cancel processing JSP or JPO by passing the parameters listed in the table below to emxIndentedTable.jsp.

These parameters are applicable only for edit mode.

URL Parameter	Possible/Default Values	Description
preProcessJPO	<JPOName>:<MethodName>	The JPO to use when a user clicks the Edit button on a structure browser component
preProcessURL	\${SUITE_DIR}/emxCustomPreProcess.jsp or ..<ApplicationDirectory>/emxCustomPreProcess.jsp Example: ../engineeringcentral/emxCustomPreProcess.jsp	The JSP called before displaying an editable structure browser page. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.
postProcessJPO	<JPOName>:<MethodName>	The JPO to use when a user clicks the Apply Edits option on a structure browser component.
postProcessURL	\${SUITE_DIR}/emxCustomPostProcess.jsp or ..<ApplicationDirectory>/emxCustomPostProcess.jsp Example: ../engineeringcentral/emxCustomPostProcess.jsp	The JSP called when a user clicks the Apply Edits option on a structure browser. The configured JSP is called after the edit processing and database update. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.
cancelProcessJPO	<JPOName>:<MethodName>	The JPO to use when a user clicks the View, Cancel, or close window button on a structure browser component
cancelProcessURL	\${SUITE_DIR}/emxCustomCancelProcess.jsp or ..<ApplicationDirectory>/emxCustomCancelProcess.jsp Example: ../engineeringcentral/emxCustomCancelProcess.jsp	The JSP called when a user clicks the View, Cancel button or closes the component window. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.

Pre Process URL for an Editable Structure Browser

The `preProcessURL` parameter specifies the name of the JSP to call during pre processing of the structure browser in edit mode.

Examples to invoke the `emxIndentedTable.jsp` with `preProcessURL`:

Set the href of any command object using macros where appropriate:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_name>
 &preProcessURL=${SUITE_DIR}/emxCustonPreProcess.jsp
```

or

Invoke `emxIndentedTable.jsp` with custom JSP pages using the relative path (macros are not supported):

```
 ../common/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_name>
 &preProcessURL=../<Application Directory>/emxCustonPreProcess.jsp
```

If both a `preProcessURL` and `preProcessJPO` are specified, the `preProcessURL` will be processed first, then the `preProcessJPO` will be processed.

The pre process JSP has the following input parameters available:

- The request parameters that were available for the `emxIndentedTable.jsp`, for example, `objectId`, `relId`, `timeStamp`, `portalMode`, `suiteKey`, etc. (`objectId` and `relId` of the root node).
- The request parameter `timeStamp` gets the information stored in the table bean as a map.
- To update or interact with any ENOVIA objects, obtain the context, from the request in the pre process JSP using this code:

```
context = (matrix.db.Context)request.getAttribute("context");
```

The following code sample shows how a custom processing page can read values from the `requestMap`:

```
<%
    // get the timeStamp from the incoming HttpServletRequest
    String timeStamp = (String)
emxGetParameter(request,"timeStamp");
    // define the indented table bean
%>
<jsp:useBean id="indentedTableBean"
class="com.matrixone.apps.framework.ui.UITableIndented"
scope="session"/>
<%
    // get the tableDataMap and the requestMap from the indented
table bean
    HashMap tableDataMap = (HashMap)
indentedTableBean.getTableData(timeStamp);
    HashMap requestMap = (HashMap)
indentedTableBean.getRequestMap(tableDataMap);
    // get specific values from the requestMap
    String languageStr = (String) requestMap.get("languageStr");
    String timeZone = (String) requestMap.get("timeZone");
    String charSet = (String) requestMap.get("charSet");
    String suiteKey = (String) requestMap.get("suiteKey");
    String stringResourceFile = (String)
requestMap.get("StringResourceFileId");
%>
```

The following code sample shows how a custom processing page can iterate through the table `objectIds`. This example 'reserves' all objects in the `objectList` (as would likely be done in a pre processing custom JSP page). Note that custom processing JSP pages do not return anything to the component.

```
<%
    // get the timeStamp from the incoming HttpServletRequest
    String timeStamp = (String)
emxGetParameter(request,"timeStamp");
    // define the indented table bean
%>
<jsp:useBean id="indentedTableBean"
class="com.matrixone.apps.framework.ui.UITableIndented"
scope="session"/>
<%
    // get the tableDataMap and the objectList from the indented
table bean
    HashMap tableDataMap = (HashMap)
indentedTableBean.getTableData(timeStamp);
```

```
MapList objectList = (MapList)
indentedTableBean.getObjectList(tableDataMap);
// loop through the objectList's list of object Maps
Iterator objectListItr = objectList.iterator();
while(objectListItr.hasNext()){
    // get the current object Map
    Map curObjectMap = (Map) objectListItr.next();
    // get the current object's objectId from the current
object Map
    String curObjectId = (String) curObjectMap.get("id");
    // a comment to pass along with the object reserve
operation
    String reserveComment = "This object is Reserved";
    // create a BusinessObject, open the object, reserve the
object, close the object
    BusinessObject curBusObject = new
BusinessObject(curObjectId);
    curBusObject.open(context);
        curBusObject.reserve(context,reserveComment);
    curBusObject.close(context);
}
%>
```

Pre Process JPO for an Editable Structure Browser

The `preProcessJPO` parameter specifies the name of the JPO program and method to invoke during pre processing.

This example invokes the `emxIndentedTable.jsp` with `preProcessJPO`:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?mode=Edit&table=<IndentedTable_name>
 &preProcessJPO=< JPO Name>:<Method Name>
```

If both a `preProcessURL` and `preProcessJPO` are specified, the `preProcessURL` is processed first, then the `preProcessJPO` is processed.

The pre process JPO specifies the `programMap` as an argument for the pre processing. The `programMap` contains the following HashMaps:

- `requestMap`--contains all the request parameters
- `paramMap`--contains these key parameters and expected values:

Parameter	Description
objectId	Object ID
relId	Relationship ID of the object

- `tableData`--contains all the column information

The above maps are packed using the `packArgs` method supported by JPO and passed to the pre process JPO being invoked. The pre process JPO can unpack this input parameter and use it for custom coding.

You can read the arguments passed for the pre process JPO as given below

```
HashMap programMap = (HashMap) JPO.unpackArgs(args);
HashMap paramMap = (HashMap) programMap.get("paramMap");
HashMap requestMap = (HashMap) programMap.get("requestMap");
HashMap tableMap = (HashMap) programMap.get("tableMap");
```

The `requestMap` contains the `languageStr`. You can read the value as shown below:

```
String languageStr = requestMap.get("languageStr");
```

The JPO can use the same method as described in [Pre Process URL for an Editable Structure Browser](#), to get the `objectList`, or it can obtain them from the `programMap` that the table component passes to the JPO. The following code sample shows how a custom processing JPO can iterate through the table `objectIds`. This example reserves all of the objects in the `objectList` (as would likely be done in a preprocessing JPO):

```
// unpack the incoming arguments into a HashMap called
'programMap'

HashMap programMap = (HashMap) JPO.unpackArgs(args);
// get the 'tableData' HashMap from the programMap HashMap
HashMap tableData = (HashMap) programMap.get("tableData");
// get the 'objectList' MapList from the tableData HashMap
MapList objectList = (MapList) tableData.get("ObjectList");
// loop through the objectList's list of object Maps
Iterator objectListItr = objectList.iterator();
while(objectListItr.hasNext()){
    // get the current object Map
    Map curObjectMap = (Map) objectListItr.next();
    // get the current object's objectId from the current
    object Map
    String curObjectId = (String) curObjectMap.get("id");
    // a comment to pass along with the object reserve
    operation
    String reserveComment = "This object is Reserved";
    // create a BusinessObject, open the object, reserve the
    object, close the object
    BusinessObject curBusObject = new
BusinessObject(curObjectId);
    curBusObject.open(context);
    curBusObject.reserve(context, reserveComment);
    curBusObject.close(context);
}
```

The pre process JPO returns a `HashMap` containing the keys defined in this table.

Key	Value
-----	-------

Action	Assigned to one of these values: CONTINUE - the process continues STOP - the process stops and the edit dialog closes. Default = CONTINUE.
Message	Either a string resource key or an original text message to display to the user. The string must use proper JavaScript format for escape characters such as single quote, double quote, and new lines.
ObjectList	The MapList of object maps that contain the editable setting value based on the reserve status.

If the value of the Action key is Continue, then the system displays the standard edit page. If the value of the Action key is Stop, then the system does not display the standard edit page.

If the Message key contains a value, it displays to the user as an alert message on top of the editable page (which is blank if the Action is Stop). When the user clicks OK in the message dialog, then the editable form page displays (if Action is Continue) or the blank form is closed (if Action is Stop).

When the blank window is closed (when Action=STOP; Message contains a value), the onUnload event (cancelProcessJPO or cancelProcessURL) should not be called.

Cancel Process URL for an Editable Structure Browser

The `cancelProcessURL` parameter specifies the name of the JSP to call during cancel processing of the structure browser in edit mode.

Examples to invoke the `emxIndentedTable.jsp` with `cancelProcessURL`:

Set the href of any command object using macros where appropriate:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_name>
 &cancelProcessURL=${SUITE_DIR}/emxCustomCancelProcess.jsp
```

or

Invoke `emxIndentedTable.jsp` with custom JSP pages using the relative path (macros are not supported):

```
 ./common/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_name>
 &cancelProcessURL=..<Application Directory>/emxCustomCancelProcess.jsp
```

If both a `cancelProcessURL` and `cancelProcessJPO` are specified, the `cancelProcessURL` is processed first followed by the `cancelProcessJPO`.

The cancel process JSP has the following input parameters available for the cancel processing:

- The request parameters that were available for the `emxIndentedTable.jsp` for example, `objectId`, `relId`, `timeStamp`, `portalMode`, `suiteKey`, etc. (`objectId` and `relId` for the root node).
- The request parameter `timeStamp` gets the table data stored in the table bean as a map.
- To update/interact with any ENOVIA objects, obtain the context from the "request" in the cancel process JSP using this code:

```
context = (matrix.db.Context)request.getAttribute("context");
```

See [Pre Process URL for an Editable Structure Browser](#) for code samples for retrieving request parameters and table data.

Cancel Process JPO for an Editable Structure Browser

The `cancelProcessJPO` parameter specifies the name of the JPO program and method to invoke during cancel processing.

This example shows how to invoke the `emxIndentedTable.jsp` with `cancelProcessJPO`:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_Name>
 &cancelProcessJPO=<JPO Name>:<Method Name>
```

If both a `cancelProcessURL` and `cancelProcessJPO` are specified, the `cancelProcessURL` is processed first followed by the `cancelProcessJPO`.

The cancel process JPO specifies the `programMap` as an argument for the pre processing. The `programMap` contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains these key parameters and expected values:

Parameter	Description
objectId	Object ID
relId	Relationship ID of the object

- `tableData` contains all the column information

The above maps are packed using the `packArgs` method supported by the JPO and passed to the cancel process JPO being invoked. The cancel process JPO can unpack this input parameter and use it for custom coding.

See [Pre Process JPO for an Editable Structure Browser](#) for code samples that retrieve request parameters and the table data.

The cancel process JPO returns a `HashMap` or returns nothing. The `HashMap` contains a single key, `Message` that contains either a string resource key or an original text message to display to the user.

If the cancel Process JPO does not need to communicate to the indented table, it can return nothing (`void`) or an empty `HashMap`. In this case, the indented table unloads the page after completing the cancel process.

Post Process URL for an Editable Structure Browser

The `postProcessURL` parameter specifies the name of the JSP to call during post processing of the structure browser in edit mode.

Examples to invoke the `emxIndentedTable.jsp` with `postProcessURL`:

Set the href of any command object using macros where appropriate:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_name>
 &postProcessURL=${SUITE_DIR}/emxCustomPostProcess.jsp
```

Or

Invoke `emxIndentedTable.jsp` with custom JSP pages using the relative path (macros are not supported):

```
 ./common/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_name>
 &postProcessURL=..<Application Directory>/emxCustomPostProcess.jsp
```

If both a `postProcessURL` and `postProcessJPO` are specified, the `postProcessURL` is processed first followed by the `postProcessJPO`.

The post process JSP has the following input parameters available for the post processing:

- The request parameters that were available for the `emxIndentedTable.jsp`, for example, `objectId`, `relId`, `timeStamp`, `portalMode`, `suiteKey`, etc. (`objectId` and `relId` of the root node).
- The request parameter `timeStamp` gets the table data that is stored in the table bean as a map.
- To update or interact with any ENOVIA objects, obtain the context from the request in the post process JSP using this code:

```
context = (matrix.db.Context)request.getAttribute("context");
```

See [Pre Process URL for an Editable Structure Browser](#) for code samples for retrieving request parameters and table data.

Post Process JPO for an Editable Structure Browser

The `postProcessJPO` parameter specifies the name of the JPO program and method to invoke during post processing.

The following example shows how to invoke the `emxIndentedTable.jsp` with `postProcessJPO`:

```
 ${COMMON_DIR}/emxIndentedTable.jsp?mode>Edit&table=<IndentedTable_Name>
 &postProcessJPO=<JPO Name>:<Method Name>
```

If both a `postProcessURL` and `postProcessJPO` are specified, the `postProcessURL` is processed first followed by the `postProcessJPO`.

The post process JPO specifies the `programMap` as an argument for the post processing. The `programMap` contains the following HashMaps:

- `requestMap` contains all the request parameters
- `paramMap` contains key parameters like `objectId`, `relId`, `languageStr`

Parameter	Description
<code>objectId</code>	Object ID
<code>relId</code>	Relationship ID of the object
<code>EditAction</code>	Assigned to one of these values: DONE - Edit dialog to close. APPLY - Default value; Edit dialog is still active and the user can continue to edit the same set of objects

- `tableData` contains all the column information

The above maps are packed using the `packArgs` method supported by JPO and passed on to the post process JPO being invoked. The post process JPO can unpack this input parameter and it can be used by the post process for custom coding.

See [Pre Process JPO for an Editable Structure Browser](#) for code samples that retrieve request parameters and the table data.

The post process JPO returns a `HashMap` or returns nothing. The `HashMap` contains the keys defined in this table:

Key	Value
Action	Assigned to one of these values: CONTINUE - the transaction is committed STOP - the transaction is aborted. Default = CONTINUE.
Message	Either a string resource key or an original text message to display to the user. The string must use proper JavaScript format for escape characters such as single quote, double quote, and new lines.

If the `Message` key contains any value, it displays to the user as an alert.

If the value of the `Action` key is `CONTINUE`, then the system commits the entire transaction of the edit process, post process JSP, and post process.

If the value of the `Action` key is `STOP` the entire transaction, including the edit process, aborts.

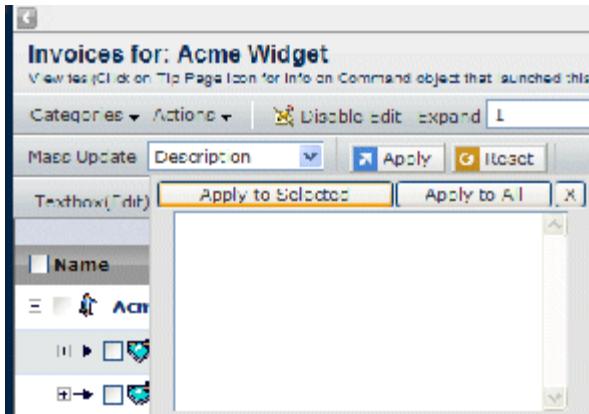
If the post Process JPO does not need to communicate to the structured browser, it can return nothing (`void`) or an empty `HashMap`. In this case, the process proceeds with the default behavior of Continue and No message.

Mass Update Toolbar

The Mass Update toolbar provides the interface to change multiple rows for a selected column with a single action.

This toolbar is included by default in edit mode, based on the property `emxFramework.ShowMassUpdate` property, which is set to true by default in `emxSystem.properties`.

The functionality and the behavior of the Mass Update toolbar is exactly the same as the configurable edit table component. The following figure shows the header section with the Mass Update toolbar.



The UI objects related to mass update are described in this table.

Action	Display	Mode	Description
View Mode (View All)	toolbar button	icon and text	This button is part of the regular page toolbar. The text and the icon for this button are changed to "View Mode" when the user clicks the "Edit Mode".
Column	option list	editable	This provides the list of columns that are editable in the current table view used in the structure browser. The user can pick any one of the columns for which mass update is required.
Value	textbox	editable	This text box provides the interface for the user to input to apply the changes. This field's input type changes, depending on the <code>Input Type</code> on the column, selected in the previous column option list.
Apply To Selected	button	button	Applies the entered value to the selected rows for the selected column.
Apply To All	button	button	Applies the entered value to all the rows for the selected column.

When entering values, the type ahead feature (see [Entering Field Values Using Type Ahead](#) for details on how this feature works).

When the user enters a value (possibly using type ahead to select a prior value), then applies the change, the JPO:method defined by the Update Program and Update Function settings for the column is invoked. The Update Program Arguments setting on the column can be used to define arguments to pass to that JPO. The structure browser uses these settings on the column to support type-ahead:

- `TypeAhead`
- `TypeAhead Program`
- `TypeAhead Function`
- `Update Program Arguments`

If the input controls for a column vary based on how the row is created or edited, then that column is not available for mass update. The values for all of these settings (if used) must be the same for that column to be available for mass update.

- `Input Type`
- `Add Input Type`
- `Lookup Input type`

For example, if a column has these settings defined, then it cannot be used for mass update:

- `Input Type=radio`

- Add Input Type=listbox

However, if all input types are defined the same (for example, Input type=radiobutton, Add Input type=radiobutton, and Lookup Input type=radiobutton), then the column can be used for mass update.

In-cell Editing

The system displays cells/headers of columns that are editable with color changes for a visual cue. To make a column editable, you must define the column with the setting `Editable=true`.

The following topics are discussed:

- [Input Controls for In-cell Edit](#)
- [Textarea Input Control](#)
- [Combo box Input Control](#)
- [Radiobutton Input Control](#)
- [Checkbox and Boolean Input Control](#)
- [Listbox Input Control](#)

Input Controls for In-cell Edit

When the user clicks on any editable field, a popup layer displays with the input field using one of these input controls (defined by the `Input Type` setting):

- textbox--This is the default. The column has a single-line box for typing text.
- textarea--The column has a multi-line box for typing text.
- radiobutton--Shows a radio button next to each range value
- checkbox--Shows a checkbox next to each range value.
- listbox--Provides a list of range values and users can select a single value. Users can select multiple values, but only one value is saved. If you implement a custom JPO, users can select multiple values and those values are saved as a comma-separated list.
- combobox--The column has a drop-down list of options and users can only select one value.

When the user clicks outside the edit field, the original cell reflects the changes, again with a visual cue that allows the user to see ongoing edits.



Textarea Input Control

This figure shows in-cell editing of a column with setting `Input Type=textarea`.

Name	Type	Description	Invoice Number	Invoice Total	Invoice Date
Acme Widget (Static)	AEFQE_SBCorpCu	A Widget a Day Keeps the Away			
Invoice AW-001	AEFQE_SBInvoice	Descr for Invoice AW-001	AW-001	1.11	11 3:05 PM
Invoice AW-002	AEFQE_SBInvoice	Descr for Invoice AW-002	AW-002	2.22	2/3/11 9:05 PM
Invoice AW-003	AEFQE_SBInvoice	Descr for Invoice AW-003	AW-003	3.33	2/4/11 3:05 AM
Invoice AW-004	AEFQE_SBInvoice	Descr for Invoice AW-004	AW-004	4.44	2/4/11 9:05 AM
Invoice AW-005	AEFQE_SBInvoice	Descr for Invoice AW-005	AW-005	5.55	2/4/11 3:05 PM

If enabled, Dynamic URLs and mxLinks can be entered into text fields. See [Dynamic URLs and mxLinks for a Column](#) for details.

If a user edits a cell, the framework validates the mxLink when the user exits the cell and displays an error message to the user.

When using the Mass Update toolbar, the framework validates the mxLink when the user clicks Apply to Selected or Apply to All in the toolbar and displays an error message to the user. For the mxLink errors using Mass Update, the user has the option to save with the errors, or to cancel and fix the problem.



Combo box Input Control

The following shows in-cell editing of a column with setting `Input Type=combo box`.

Sub Total	Tax	Paid
30.0	1.79	FALSE
14.96	0.89	FALSE
44.99	2.69	TRUE
7.99	0.47	FALSE



Radiobutton Input Control

The following shows in-cell editing of a column with setting `Input Type=radiobutton`. You can also provide a value for the `Input Control Direction` setting to define whether the values show horizontally or vertically (default).

Append Columns

Append Column Date	Append Column rad
1/29/11 12:06 AM	<input type="radio"/> 1.1 <input type="radio"/> 2.2 <input type="radio"/> 3.3 <input type="radio"/> 4.4 <input type="radio"/> 5.5 <input type="radio"/> 6.6
1/29/11 6:06 AM	
1/29/11 12:06 PM	
1/29/11 6:06 PM	
1/30/11 12:06 AM	



Checkbox and Boolean Input Control

The following shows in-cell editing of a column with setting `Input Type=checkbox`. You can also provide a value for the `Input Control Direction` setting to define whether the values show horizontally or vertically (default).

Append Column CheckBox	Append Column List
<input type="checkbox"/> 1.1 <input type="checkbox"/> 2.2 <input type="checkbox"/> 3.3 <input type="checkbox"/> 4.4 <input type="checkbox"/> 5.5 <input type="checkbox"/> 6.6	

For fields (shown as columns) that have boolean values, a single checkbox shows in the Edit mode for that cell. The field is considered to be boolean if:

- The expression for the object or relationship resolves to an attribute of type Boolean.
- The expression for the object or relationship resolves to an attribute of type String that contains two ranges values specified in the `emxFramework.UIFreezePane.Checkbox.BooleanValues` property (see the *Live Collaboration Administrator's Guide* for details).
- A Range Program and Range Function setting evaluates to two range values listed in the above-specified property.



Listbox Input Control

The following shows in-cell editing of a column with setting `Input Type=listbox`. You can also define a value for the List Box Selection setting to define whether the user can select a single or multiple (default) values.

Append Column ListBox
<input type="listbox"/> 3.3 4.4 5.5 6.6 7.7

Markup Interface

The structure browser can display, edit, and save a markup. The structure browser provides the hooks for these tasks; you must define the custom page that implements the markup.

The JavaScript API allows the structure browser to:

- Display 1 or more markups for a specific object, including visual cues
- Send markup data for a single level from a selected object
- Save the markup

When saving a markup, a single markup is created for each parent object that had changes. Saving a markup saves all changes for all objects in the structure browser regardless of row selection.

See "About the Structure Browser" in the *Application Exchange Framework User's Guide* for a description of the visual cues.

The JavaScript sends and loads XML string data; the custom application must handle saving the markup in the database.

Once loaded (the custom application passes the XML string data to the structure browser), users can work with the data as in any other structure browser, including any editing functions configured for the page. If the `showApply=true` URL parameter was passed to the structure browser, the edit mode includes an Apply button. If the user clicks this button, the edits, including any loaded markups, are saved to the database. To prevent the user from saving marked-up changes to the database, pass the `showApply=false` URL parameter. The custom application must then provide a means, such as saving the markup, to retain the edits made by the user.

The API includes these JavaScript methods to work with markups:

- `editableTable.getMarkupXML()`--Returns an XML string to the custom page.
- `editableTable.loadMarkupXML(URL - JSP with parameters)`--Applies a markup to an already loaded structure browser page.

Refer to the API documentation for details about the JavaScript methods to perform this functionality.

Editable Columns

The structure browser supports the entire functionality of the editable configurable table component. All the settings and parameters used by the editable configurable table are applicable.

See [Parameters for Table Objects](#) and [Settings for Table Column Objects](#).

You must define the `Field Type` setting on any column defined as editable. Otherwise, the user can change the value in that column, but their edits will not be saved.

You can use a JPO:method to control edit access to individual cells within a column. Use the `Edit Access Program` and `Edit Access Function` settings on the column to specify the needed JPO and method. The structure browser passes the `columnMap`, `requestMap` and `objectList` to the JPO. The method returns a `StringList` of boolean values for each cell (true/false) defining the edit access to that cell.

Adding Rows for New or Existing Objects to the Structure

You can define a structure browser to use toolbar buttons to add new rows (for new objects or to connect existing objects) to the structure browser.

The following topics are discussed:

- [Parameters for Adding Rows](#)
- [Location of Inserted Rows](#)
- [Columns for In-Line Creation/Connection](#)
- [Add a New Object](#)
- [Connect an Existing Object \(Lookup\)](#)
- [Apply Modifications to a Structure Browser](#)

Parameters for Adding Rows

These URL parameters are used:

- `lookupJPO`. Defines the JPO:method used to find the needed object to connect, based on search criteria entered by the user. The user clicks the **Lookup Entry** button to invoke the JPO.
- `addJPO`. Defines the JPO:method used to create the object in the database from user-entered values. The user clicks the **Apply** button to invoke the JPO.
- `applyURL`. Defines the URL to which all newly-added and modified rows are posted as an XML element with the parameter name `data`. The user clicks the **Apply** button to submit the defined URL.

If both the `applyURL` and `addJPO` URL parameters are passed to the structure browser, the `addJPO` parameter is ignored.

The structure browser uses to indicate a new row used to create a new object, and to indicate a new row for connecting an existing object

Name	Type	Description	Cust ID	Order ID	Order Date	Some Column
Acme Widget	AEFQE_S	A Widget a Day Keeps the IRS Away	12			
Order AW-001	AEFQE_S	Descr for Order AW-001	12	AW-001	2/2/11 12:06 AM	
Order AW-002	AEFQE_S	Descr for Order AW-002	12	AW-002	2/2/11 6:06 AM	

When creating a new row, if the attribute shown in a column has a schema-defined default value, that value is automatically populated in the column. However, if that column has the `Default Program` and `Default Function` settings defined, the value returned by the `Default_AddNewRow` or `Default_ExistingRow` key from the JPO overrides the schema-defined default value.

See the *Application Exchange Framework User's Guide* for details on how the user works with this tools.

The display of these toolbar buttons is controlled by which URL parameters are passed to the structure browser:

- (create new object) displays if `addJPO` or `applyURL` is passed
- (connect existing object) displays if `lookupJPO` is passed
- (remove inserted row) displays if any of these parameters is passed: `addJPO`, `applyURL`, or `lookupJPO`

If the indicated URL parameter is not passed, then that toolbar button does not display on the structure browser.



Location of Inserted Rows

When using or , the location of the inserted row depends on whether the structure browser is defined to use flat table mode, single root object, or multiple root objects.

When inserting a row in a structure browser in flat table mode (see [Structure Browser in Flat Table Mode](#)), the row is inserted at the top of the structure browser. The user cannot change this location by selecting any rows in the structure browser.

When inserting a row in a structure browser defined with a single root object, the placement depends on whether any rows

are selected:

- If no row is selected, the new row is added to the top of the first level of children (the level immediately below the root node).
- If one row is selected, the new row is added as a child to the selected row, at the top of that node's list of children.
- If more than one row is selected, an error message displays.

When inserting a row in a structure browser defined with multiple root nodes, the placement also depends on whether any rows are selected:

- If no row is selected, the new row is added as a root node at the top of the listing for the page.
- If one row is selected, the new row is added as a child to the selected row, at the top of that node's list of children.
- If more than one row is selected, an error message displays.

The  (Remove) button can be used to remove any selected rows inserted using  or  prior to the **Apply** button being clicked.



Columns for In-Line Creation/Connection

To enter data inline for a new object (see [Add a New Object](#)) or for connecting an existing object (see [Connect an Existing Object \(Lookup\)](#)), these settings for the column are used:

- **Lookup Input Type**. The input control used to enter search criteria when connecting an existing object.
- **Add Input Type**. The input control used to enter values when creating a new object.
- **autoNumber**. Used for Name columns. When autonaming is used for objects, this setting allows the user to choose the auto number series. The value for this setting is the symbolic name for the type, for example:
`autoNumber=type_HardwarePart`.

The **Lookup Input Type** and **Add Input Type** settings can take any of these values:

- `textbox` (default)
- `combobox`
- `radiobutton`
- `listbox`
- `list box with free form entry`

These input controls work the same as described in [In-cell Editing](#).

The list box with free form entry combines the listbox and textbox input controls. The range values defined for the attribute show as a list of check boxes, and the last check box is for a text box. If that option is checked, the user must enter text in the box.

As with other list boxes, if the user selects multiple values (using the control key), only one value is saved unless you define a custom JPO to process multiple values.

If the `autoNumber` setting is defined for a column (usually a Name column), the user can select a series or manually enter a name.



For structure browsers that may have different object types in a column, the `OnChange Handler` or `OnFocus Handler` settings must be used to select the appropriate autonumbering for the current row.



Add a New Object

Users can add a blank row to the structure browser to create a new object. This new row is equivalent to creating a new object using a create form. To support this function, the structure browser must display all columns that are defined as required on the equivalent create form. When the user adds a row, the structure browser automatically populates any fields defined with default values. The user can change any of these values as needed.

After the user enters the data for the new object, the **Apply** button invokes the JPO:method specified by the `addJPO` URL parameter to create the object. The user can add any number of rows prior to clicking the **Apply** button.

The page passes all newly-created rows to the JPO as input parameters. The JPO returns a list of the created objects. If the data entered for any row did not define a unique object (or another error occurs), the JPO returns an error. The structure browser indicates the error (user can see text of message by mousing over the ).

Name	Type (TextBox)	Status	Description (TextArea)	Cust ID (
Acme Widget	AEFQE_SBCorpCust	Active	A Widget a Day Keeps the IRS Away	12
Invoice AW-002	AEFQE_SBInvoice	Create	testqwqe	12
Invoice AW-002	AEFQE_SBInvoice	Create	Cannot create an object, already object with this Type & Name exists	
Invoice AW-002	AEFQE_SBInvoice	Create	Descr for Invoice AW-002 werwe werre	

If both the `applyURL` and `addJPO` URL parameters are passed, the `addJPO` parameter is ignored.

See the JavaDocs for details of the JPO.



Connect an Existing Object (Lookup)

Users can also add a blank row used to load an existing object from the database. The user enters search criteria in the row, then clicks **Lookup Entries**. The system first validates the criteria (using the validation method defined by the `OnChange` Handler setting on the column), then invokes the JPO defined by the `lookupJPO` URL parameter passed to the structure browser.

The `OnChange` Handler for the column is invoked with these parameters:

- cell value
- `New` or `Lookup`

For the second parameter, `Lookup` is used for inserted rows and `New` is used for the normal apply function.

The page passes the search criteria entered in the added rows to the JPO as input parameters. The JPO returns the matching objects, or an error message if more than one or no matching object was found. The structure browser indicates the error (user can see text of message by mousing over the ).

Name	Type (TextBox)	Status
Acme Widget	AEFQE_SBCorpCust	Active
Invoice AW-001	AEFQE_SBInvoice	Create
Invoice AW-02	AEFSB_SBInvoice	Active
Invoice AW-02	AEFSB_SBInvoice	No object found!
Order AW-001	AEFQE_SBOOrder	Open

If a column does not have the `Lookup Input Type` setting defined, the user may still enter a value although it is not used as search criteria. When the object is found, if the database value is different from the user-entered value, the cell shows as though the user edited the value: the original database value is shown in red, strikethrough text, with the value the user entered shown in bold italic text. This also applies for columns populated based on a relationship to the object (not as an attribute of that object).

In addition, if a column does not have the `Lookup Input Type` setting defined but has the `Default Program` and `Default Function` settings defined, the value returned by the `Default_AddNewRow` or `Default_ExistingRow` key from the JPO shows as though the user had entered that value. That is, if the database value is different from the user-entered value, the database value is shown in red, strikethrough text and the user-entered value is shown in bold, italic text.

Once all rows inserted using  have been verified and the page is updated by the JPO:method defined by the `lookupJPO`, the button on the structure browser changes to **Apply**.

See the JavaDocs for details of the JPO.



Apply Modifications to a Structure Browser

When the user clicks the **Apply** button, the structure browser either:

- invokes the JPO:method specified by the `addJPO` URL parameter
- submits the URL defined by the `applyURL` parameter

If both the `applyURL` and `addJPO` URL parameters are passed to the structure browser, the `addJPO` parameter is ignored and the `applyURL` is submitted. The URL can be any configurable component or custom JSP page. The newly-added and modified rows are submitted as an XML element with the parameter name `data` to the specified URL. The posted data can be accessed from the specified URL using request parameters.

If the URL specifies a custom JSP page, that page should invoke the API `emxEditableTable.refreshStructureWithOutSort()` to refresh the view (see [JavaScript APIs](#)). If an error occurs in the JSP page, the custom page should invoke the API `emxEditableTable.displayValidationError()`; with an XML containing the `rowId` and error message. See [Displaying Validation Errors](#).

When the `applyURL` parameter is passed, and defined `postProcessURL` or `postProcessJPO` are not invoked.

Validating Cell Values

You can use JavaScript methods to validate any cell when the structure browser is displayed in Edit mode and the Editable setting is true.

Typically, validation is initiated using the `OnChange Handler` setting defined for a column. The `emxUIFreezePane.js` file contains these methods that can be used for validation:

- `emxEditableTable.getCellValueByRowId`: Returns an object that holds the actual/display values for both the modified and old values of the cell by referencing the rowID and column name.
- `emxEditableTable.getCellValuebyObjectRelId`: Sets the displayed field value and actual field value by referencing a relId, objectId, and column name.
- `emxEditableTable.setCellValueByRowID`: Sets the actual/display value of the cell identified using the rowId and column name.
- `emxEditableTable.setCellValueByObjectRelId`: Sets the actual/display value of the cell identified using the relId, objectId, and column name
- `emxEditableTable.setEditableByRowId`: Enables or disables edit mode for a cell identified using the RowId and column name.
- `emxEditableTable.setEditableByObjectRelId`: Enables or disables edit mode for a cell identified using the RelId, ObjectId, and column name.
- `emxEditableTable.getCurrentCell`: Returns details about the current cell.
- `emxEditableTable.getParentRowId`: Returns the RowId of the parent row for the current cell.
- `emxEditableTable.getChildrenRowIds`: Returns the RowIDs of the first-level children for the current cell.
- `emxEditableTable.reloadCell`: If the column has the Reload Program and Reload Function settings defined, calls that method to reload the cell value.

See [JavaScript APIs](#) for details on these functions.

Custom Client-side Validation

The structure browser component provides the necessary interface to implement custom client-side validation on the editable column cells. You must define the custom JavaScript method call for validating any cell update in a JS or JSP file in the suite-specific directory.

This file must be a pure JavaScript file and should not contain any HTML to JavaScript tags. Only JS variables and methods may be defined in these files.

This file can include a function that refreshes the structure browser. This function is specified using the `onReset` URL parameter. The function is executed after resetting the values in the edit page, and before refreshing the page.

You can define one or more validation files specific to the structure browser under the suite-specific property in `emxSystem.properties`.

```
eServiceSuite<Suite Name>.UIFreezePane.ValidationFile
```

As an example, for Engineering Central the property is:

```
eServiceSuiteEngineeringCentral.UIFreezePane.ValidationFile =  
emxEngFormValidation.jsp
```

You must configure the method to invoke for validating any cell as the column setting `OnChange Handler`. Assign it to the method name without parentheses. For example, if the validation method is `checkUniqueName()`, then add the following setting:

```
OnChange Handler = checkUniqueName
```

The system passes the validation routine with an argument[0] that is the value assigned to the new value entered by the user. The validation routine can also leverage the JavaScript APIs (listed in the following section) for getting the details of the related fields in the structure.

The structure browser in edit mode supports client-side custom validation for two different events. It uses these settings: `OnChange Handler` and `ValidateOnApply`.

Assign these settings to the JavaScript method name (without parenthesis) and define the method in one of the JavaScript custom validation files.

- `OnChange Handler` Validates changes from in-cell editing interface.
- `ValidateOnApply` Validates changes upon clicking on the Apply Edit link.

When using the `OnChange Handler` setting in a structure browser with expanded rows, the change may cascade up the hierarchy requiring changes to other cells. If the structure browser has several levels expanded and/or many columns, the process may take several seconds. A message displays to notify the user that the changes are being processed.

JavaScript APIs Available for Validation Methods

Business Process Services provides APIs for validation methods.

The APIs available to the validation methods are:

```
getValueForColumn("<column name>")
```

Use this method to get the cell value for a given column cell in the current row being edited by the user. For example, if the user is editing the column cell Quantity at row 10, then to get the value of another column cell Name at row 10, call the API as:

```
getValueForColumn("Name");  
getColumnDataAtLevel()
```

Use this method to get all the cell values in the current level under the immediate parent node.

```
getColumnDataAtLevel();
```

XHTML Standard

Since the structure browser is driven by XML, the only way to insert HTML into the document is to insert it in the XML. Because of this, programHTML has to conform to the XHTML standard.

The following lists the most important changes to make.

- All tags need to be closed.

For example (note the xml style closing backslash ^/>^):

```
RefDes = ^<img src='..../common/images/iconStatusAlert.gif' border='0' alt='^+incompatibleFormatAlt+^' title='^+incompatibleFormatAlt+^' />&#160;^+RefDes;
```

- Use instead of &nbs;
- In URLs the & character will cause an error. Use &

For example:

```
columnVals.addElement('^<a href='^javascript:emxTableColumnLinkClick('..//engineeringcentral/emxpathEditEBOMDialogFS.jsp?emxSuiteDirectory=engineeringcentral&relId='^+relId+'^&suiteKey=EngineeringCentral&parentOID='^+parentId+'^&objectTypeId='^+id+'^', '700', '600', 'false', 'popup', '')^><img border='0'^ src='^images/iconActionEdit.gif'^ alt='^'^/></a>^');
```

- All attributes need quotes.

```
src='..../common/images/iconStatusAlert.gif' border='0'
```

Or

```
border='0'^ src='^images/iconActionEdit.gif'^
```

- All images must have a height attribute of NOT more than 16.

```
<img height='16'^>
```

Structure Browser in Flat Table Mode

You can configure a structure browser page to display as a flat table, that is, the nodes cannot be expanded/collapsed. This feature allows the page to take advantage of functionality provided by the structure browser that is not available with the normal table pages, such as the freeze pane.

The following topics are discussed:

- [Specifying Flat Table Mode](#)
- [Identifying Table Rows](#)
- [URL Parameters Supported/Not Supported in Flat Table Mode](#)

Specifying Flat Table Mode

Flat table mode is used to display the structure browser when the `program` URL parameter is passed, but none of the expand parameters (`relationship`, `expandProgram`, or `expandProgramMenu`) are passed to `emxIndentedTable.jsp`. This screen shot shows an example of a structure browser page in flat table mode.

Name	Type (TextBox)	Cust ID (ComboBox)	Street Addr (TextBox)	City (CheckBox)	State (TextBox)
Acme Widget	AEFQE_SBCorpCustomer	12	167 Widget Way	Mumbai	CT
Amalgamated Broom	AEFQE_SBCorpCustomer	45	2525 Sweep Street	Pune	CT
Zirconium Keg	AEFQE_SBCorpCustomer	678	1967 Invention Lane	New York	MT
ZZ Tops & Novelties	AEFQE_SBCorpCustomer	12	1973 La Grange	New York	TX

In flat table mode, the standard Type filter, Relationship filter, Direction filter, and Expand filters do not display, even if specified.

Identifying Table Rows

In flat table mode, each row is identified using the form element `emxTableRowId`:

```
"<relationshipid>|<objectId>|<parent objectId>|<level id>"
```

Rows in flat table mode likely will not have a relationshipid, parent object, or level id, so an example row would look like:

```
"| 48634.14845.49563.55204| |"
```

The pipe delimiters are placeholders for the unavailable ids.

URL Parameters Supported/Not Supported in Flat Table Mode

Flat table mode of the Structure Browser supports these URL parameters (see [URL Parameters Accepted by emxIndentedTable.jsp](#) for parameter details):

- cancelLabel
- editLink
- export
- freezePane
- header
- helpMarker
- massUpdate
- mode
- objectId
- printerFriendly
- program
- selection
- sortColumnName
- sortDirection
- subHeader

- SubmitURL
- submitLabel
- table
- tableMenu
- TipPage
- toolbar
- TransactionType

These URL parameters are NOT supported in flat table mode:

- direction
- directionFilter
- expandFilter
- expandProgram
- expandProgramMenu
- relationship
- relationshipFilter
- type
- typeFilter

Filter for a Structure Browser Page

You can provide an auto filter mechanism for root nodes of a structure browser page or for a structure browser shown in flat table mode the same way as for a table page.

See [Structure Browser in Flat Table Mode](#). The auto filter does not work for structure browsers with a single root node, and for multiple root nodes only filters on the root nodes, not any expanded rows. For more information on table filtering, see [Building a Table](#).

See the *Application Exchange Framework User's Guide* to see how this feature works.

The filter icon, , is automatically added to the page toolbar if the structure browser includes any column that:

- Is listed in the `emxFramework.AutoFilter.Basic.InclusionList` property in `emxSystem.properties`
- Is listed in the `emxFramework.AutoFilter.Attribute.InclusionList` property in `emxSystem.properties`
- Has ranges defined for the attribute
- Has the `Auto Filter=true` setting defined (cannot use with a column that can have multiple values for a single cell)

Even if any of the above criteria are met, the filter icon does not displays if the `autoFilter=false` URL parameter is passed.

If you want to exclude attributes with defined ranges from being used as filters, add the attribute to the `emxFramework.AutoFilter.Attribute.ExclusionList` property in `emxSystem.properties`.

If the column contains an attribute configured with a dimension, the page is filtered based on the display value, not the normalized value.

Configuring many columns for autofiltering or applying an autofilter to inappropriate columns can adversely effect performance when viewing the filtered list. For optimum performance, only define a limited number of columns with autofilter. Choose columns that have a limited set of possible values on which the user may want to filter the table data. For example, state and owner columns are good candidates. Columns such as description and name are inappropriate because of the unlimited number of resulting values.

Calculations for the Structure Browser

You can configure a structure browser to provide calculations for numeric columns.

In this section:

- [About Column Calculations](#)
- [Calculation Settings](#)
- [Decimal Precision](#)
- [Custom Calculations](#)
- [Arithmetic Calculations](#)

About Column Calculations

You can configure a structure browser to provide calculations for numeric columns.

Calculations are available dynamically--using Σ in the toolbar--or you can configure them to automatically provide calculations for columns. For information on dynamic calculations, see "Performing Column Calculations" in the *Application Exchange Framework User's Guide*.

The calculation section header shows a count of the number of objects. You can use column values to show total, average, maximum, minimum, median, and standard deviation values. You can apply one or more calculation types to each column. The results display at the bottom of the structure browser.

The screenshot shows a Structure Browser window titled "Orders placed by Acme Widget". The main area displays an "Orders Table" with the following data:

Name	Type	Description	Cust ID	Order ID	Order Date	Sub Total	Tax	Paid
Order AW-001	AEFQE_SBOOrder	Descr for Order AW-001	12	AW-001	2/3/11 3:05 PM	30.0	1.79	FALSE
Order AW-002	AEFQE_SBOOrder	Descr for Order AW-002	12	AW-002	2/3/11 9:05 PM	14.96	0.89	FALSE
Order AW-003	AEFQE_SBOOrder	Descr for Order AW-003	12	AW-003	2/4/11 3:05 AM	44.99	2.69	FALSE
Order AW-004	AEFQE_SBOOrder	Descr for Order AW-004	12	AW-004	2/4/11 9:05 AM	7.99	0.47	FALSE
Order AW-005	AEFQE_SBOOrder	Descr for Order AW-005	12	AW-005	2/4/11 3:05 PM	0.0	0.0	FALSE

Below the table, there is a summary row:

All (5 items)						97.94	5.84	
Total						44.99	2.69	
Maximum								

At the bottom of the browser, there is a toolbar with various icons, including a Σ icon. The status bar shows "Page 1 of 1".

If you pass the `calculations=false` URL parameter but the structure browser includes a column that contains the calculate setting set to true, then the calculations display but the toolbar icon does not.

These rules apply to the calculations:

- The column name cannot contain a space character.
- You can perform calculations only on numerical values. To perform calculations on a column of string attributes that have numerical values, set `format=numeric` in the column. Calculations are also available for columns with the `format=arithmetic` defined.
- Null or empty rows are considered as a value of zero. This allows you to perform calculations on data sets that may be missing values for some rows.
- You can perform more than one calculation on the same column. For example, a column may have a sum and an average specified for it. This results in two rows in the calculation section, one for each.
- Place the sum calculation as the first calculation in the section at the bottom of the page. This prevents confusion about what the total represents.
- By default, Σ does not display in the toolbar (`calculations = false`). You can show the button by passing `calculations = true` to `emxIndentedTable.jsp`.

If the column contains data for an attribute configured with a dimension, the framework uses the normalized values (the values stored in the database) to perform and display the calculations.

Calculations show in both View and Edit mode. In Edit mode, calculations are immediately updated when values in the numeric column are changed.

Calculation Settings

You can define the following settings for any numerical column in the structure browser. Labels display in the first column of the listing that is not a checkbox, a radio button, or an icon image. The label column spans the remaining columns up to the first column that contains a calculation.

Setting	Description	Values/Examples
Arithmetic Expression	Comma-separated list of arithmetic expressions. Expressions can include cell variables. See Arithmetic Calculations for details.	
Calculate Average	When true, calculates the average of values in the column, using the defined decimal precision. By default, the calculation uses the system-wide precision (set in emxSystem.properties), or you can use the Decimal Precision setting to set a specific precision for this column calculation.	true false (default)
Calculate Average Label	When the Calculate Average setting is used, any value for this setting shows as the label for the column average. If not defined, "Average" is used as the label.	Average (default) <any string value>
Calculate Custom	When true, invokes the program defined by the Calculate Custom Program setting.	true false (default)
Calculate Custom Label	When the Calculate Custom and Calculate Custom Program settings are used, any value for this setting shows as the label for that calculated value. If not defined, "Custom Label" is used as the label.	Custom Label (default) <any string value>
Calculate Custom Program	If the Calculate Custom setting is true, this setting defines the program and function to invoke. The program can perform any calculations needed for the data in the column, with a single result.	program:function
Calculate Maximum	When true, displays the largest value of the column values.	true false (default)
Calculate Maximum Label	When the Calculate Maximum setting is used, any value for this setting shows as the label for the column maximum. If not defined, "Maximum" is used as the label.	Maximum (default) <any string value>
Calculate Median	When true, displays the middle value (the mean) of the column values. If the column contains an even number of values, the average of the 2 middle numbers is used, using the defined Decimal Precision. By default, the calculation uses the system-wide precision (set in emxSystem.properties), or you can use the Decimal Precision setting to set a specific precision for this column calculation.	true false (default)
Calculate Median Label	When the Calculate Median setting is used, any value for this setting shows as the label for the column median. If not defined, "Median" is used as the label.	Median (default) <any string value>
Calculate Minimum	When true, displays the smallest value of the column values.	true false (default)
Calculate Minimum Label	When the Calculate Minimum setting is used, any value for this setting shows as the label for the column minimum. If not defined, "Minimum" is used as the label.	Minimum (default) <any string value>
Calculate Standard Deviation	When true, displays the standard deviation of the column values using this formula:	true false (default)

$$sd = \sqrt{\frac{\sum (x - \bar{x})^2}{N-1}}$$

where:

N=total number of elements

\bar{x} =mean of the column values

By default, the calculation uses the system-wide decimal precision (set in emxSystem.properties), or you can use the Decimal Precision setting to set a specific precision for this column calculation.

Calculate Standard Deviation Label	When the Calculate Standard Deviation setting is used, any value for this setting shows as the label for the column calculated value. If not defined, "Standard Deviation" is used as the label.	Standard Deviation (default) <any string value>
Calculate Sum	When true, displays the total of the column's values.	true false (default)
Calculate Sum Label	When the Calculate Sum setting is used, any value for this setting shows as the label for the column sum. If not defined, "Total" is used as the label.	Total (default) <any string value>
Decimal Precision	Defines the decimal precision for all calculations for this column. The system properties setting <code>emxFramework.SBTableCalculations.DecimalPrecision</code> defines the system-wide value; this setting overrides that value. You only need to use this setting if you want to use a value other than the system-wide value (defined in emxSystem.properties).	Any positive integer.

Decimal Precision

For total, maximum and minimum table column calculations, the system does not round the value. Average, median and standard deviation calculations involve division of numbers so rounding is done when necessary.

In emxSystem.properties, the property `emxFramework.SBTTableCalculations.DecimalPrecision` defines how many digits follow the decimal point when rounding. By default, the calculated value is rounded to 5 digits after the decimal point if the result contains more than 5 digits after the decimal point. For example, if the calculated value is 25.1234562, then the value would be rounded to 25.12346.

You can override the system default by defining the Decimal Position setting for a specific column used in the structure browser.

Custom Calculations

You can use the [Calculate Custom](#) and [Calculate Custom Program](#) settings on a column to define calculations not covered by any of the other settings

You should also define a value for the [Calculate Custom Label](#) to indicate what the shown value represents. Otherwise, "Custom Label" shows, and users will not know what that means. The structure browser invokes the indicated program, and the program returns a string value. That value is displayed in the calculations section beneath the column defined with the setting

Arithmetic Calculations

You can construct arithmetic equations that use values in other columns in the structure browser.

In addition to the calculations defined by the settings listed in [Calculation Settings](#), you can also construct arithmetic equations that use values in other columns in the structure browser.

The following topics are discussed:

- [Variables](#)
- [Syntax](#)
- [Examples](#)

Variables

You perform arithmetic calculations using variables that define the column to use and the level (of expansions). Variables use this format:

`<ColumnName>[<Level>]`

You can use any column name in the structure browser. When defining a level:

- The value must be $=> 0$; use 0 for root level objects.
- To identify all cells in the column, use `CURRENT` (indicates the level of the cell on which the expression is being evaluated).
- Use `LEAF` to indicate the last level in the structure.
- You can include expressions such as `CURRENT+1` or `LEAF-1`.
- You can include a comma-separated list of levels, such as `<ColumnName>[0,1,2,3]` or use the not equal operator such as `<ColumnName>[!5]`.
- Level defaults to `CURRENT` if not specified.

You can define the label used for aggregate functions, such as SUM or AVG, using `<ColumnName>["Label"]`.



Syntax

An arithmetic calculation uses this syntax:

`<LHS>=<RHS>`

LHS is Left Hand Side; RHS is Right Hand Side.

RHS is optional; if not provided it defaults to the variable `<ColumnName>[CURRENT]`, or `<ColumnName>`. When used, RHS must be a cell variable and defines where to put the calculated variable in the structure browser.

LHS can use any arithmetic operations (+ - * / %), arithmetic functions, or aggregation functions and can reference other cell variables. The arithmetic functions include all those supported by the JavaScript Math class. In addition, you can use the string function `Concat`.

`Concat("Bal: ", Jan + Feb + Mar)`

The function can call any string or arithmetic expressions.

This table lists the supported aggregation functions:

Function	Description	Syntax/Examples
SUM	Totals the specified cells.	<code>SUM(Cost[0])</code> <code>SUM(Shipping Cost[0]),Labor Cost[0])</code>
AVE	Average of the specified cells.	<code>AVE(Cost[0])</code> <code>AVE(Shipping Cost[0]),Labor Cost[0])</code>
MAX	Maximum value in the specified cells.	<code>MAX(Cost[0])</code> <code>MAX(Shipping Cost[0]),Labor Cost[0])</code>
MIN	Minimum value in the specified cells.	<code>MIN(Cost[0])</code> <code>MIN(Shipping Cost[0]),Labor Cost[0])</code>
MEDIAN	Median value in the specified cells.	<code>MEDIAN(Cost[0])</code> <code>MEDIAN (Shipping Cost[0]),Labor Cost[0])</code>



Examples

Say you want to calculate a Session Total, which is the sum of the values for the columns with names Jan, Feb, and Mar. You could perform this calculation using this expression:

```
"Jan[CURRENT] + Feb[CURRENT] + Mar[CURRENT]"
```

or

```
Session Total[CURRENT] = "Jan[CURRENT] + Feb[CURRENT] +  
Mar[CURRENT]"
```

The results of the arithmetic expression are shown in the Session Total column.

This next example uses parent/child nodes. A structure browser page includes an Organization that can be expanded to show Companies within that organization, and each Company can be expanded to show the Business Units within the company. You can total the cost of all children objects using this expression:

```
Cost = SUM(Cost[CURRENT+1])
```

or

```
Cost[CURRENT] = SUM(Cost[CURRENT+1])
```

For each company, the Cost column shows the total for all business units; for each organization, the Cost column shows the total for all companies.

To prevent the calculation from being applied to business units (which have no children in this example), you can use this expression:

```
Cost[0,1] = SUM(Cost[CURRENT+1])
```

or

```
Cost[CURRENT, !LEAF] = SUM(Cost[CURRENT+1])
```

The first expression indicates to execute the expression for all root (organization) and their children (companies), and for no further levels. The second expression indicates to execute the expression for all levels except the leaf (final) level.

Structure Compare

The emxStructureCompare.jsp and emxStructureCompareReport.jsp programs provide a tool to compare 2 objects in a structure browser or table page. The objects can be selected in the page or selected in the criteria dialog box. To include this functionality, you need to add the AEEFStructureCompare command to the toolbar for the page. The compare function can be initiated from either View or Edit mode of the structure browser.

In this section:

- [Default Structure Compare Dialog Box](#)
- [Column Settings for Structure Compare](#)
- [Custom Structure Compare Dialog Box](#)
- [Structure Compare Report](#)
- [URL Parameters Accepted by emxStructureCompare.jsp and emxStructureCompareReport.jsp](#)

Default Structure Compare Dialog Box

This section defines the contents of the default dialog box for entering data to generate a structure compare report.

The default dialog box called by the AEFStructureCompare command for selecting comparison criteria is:

Structure Compare Criteria

Fields in red italics are required

Object 1	<i>AEFQE_SBCorpCustomer</i> Acme Widget
Object 2	<i>AEFQE_SBCorpCustomer</i> Amalgamated Broom
Match Based On	<input type="button" value="Name"/> <input type="button" value="None"/> <input type="button" value="None"/>
Compare By	<input type="checkbox"/> Name <input type="checkbox"/> Type <input type="checkbox"/> Revision <input type="checkbox"/> Description <input type="checkbox"/> Cust ID <input type="checkbox"/> Street Addr <input type="checkbox"/> City <input type="checkbox"/> Zip Code <input type="checkbox"/> Country <input type="checkbox"/> Status
Expand Level	1
Compare Mode	<input checked="" type="radio"/> Complete Summary <input type="radio"/> Difference Only Report <input type="radio"/> Unique to Left Report <input type="radio"/> Unique to Right Report <input type="radio"/> Common Components Report
<input checked="" type="checkbox"/> Done <input type="checkbox"/> Cancel	

If the user selected no more than 2 rows prior to clicking Structure Compare in the toolbar, the names of those objects are pre-populated in the Object 1 and Object 2 fields. The user can also use the browse button to use the Framework general search dialog to select a different object for either field.

The Match Based On section provides 3 field choices for the user to select columns used to indicate a match. To be considered a match, the values for all 3 fields must be the same for both objects. The user can select 1, 2, or 3 fields. The combo boxes list all fields used in the table except those with a *Column Type of Image, Icon or Separator*. To eliminate a column from these lists, you need to set the *Comparable=false* setting for that column.

The Compare By section lists all fields that have the *Comparable=true* setting defined. Fields selected for "Match Based On" are not selectable for this section. When the compare function finds child objects of Object 1 and Object 2 that are considered matches (the values for columns selected in Match Based On section are the same), then the columns selected in this section are used to compare the two objects.

The Expand Level determines the initial number of expansions before processing the compare logic.

Compare Mode defines the appearance of the compare report, whether it includes the entire summary of objects (Complete Summary), or only listing the differences between 2 objects (Difference Only Summary).

Column Settings for Structure Compare

By default, all columns in a structure browser (except those with column types of Image, Icon, or Separator) are shown in the compare report and are available for the user to select as a basis of comparison. You can use column settings to override the default value.

The `Compare Report` column setting can be set to `hide` (default value is `show`) to remove that column from the structure compare report.

The `Comparable` column setting can be set to `false` (default value is `true`) to remove that column from the list of Compare By columns.

The `Diff Code` column setting defines the value to show in the Diff Code column for the Complete Summary report. This column lets you see how the objects differ. The syntax for this setting is:

`Diff Code = <precedence:code value>`

See [Complete Summary Report](#) for details on using this column setting.

Custom Structure Compare Dialog Box

You can customize the structure compare dialog box. You need to write a custom JSP for this purpose. The custom dialog page must post the mandatory criteria fields and needed URL parameters to the structure compare report generator (emxStructureCompareReport.jsp).

This table lists the fields/parameters that can be passed:

HTML Form Element/URL Parameter	Description	Mandatory
compareBy	1 or more columns used to compare matched objects. The column names can be: Passed as a comma-separated list in the URL Parameter compareBy Posted as an HTML form element with the element name compareBy. Each form element must have this name and contain the unique values for that specific match.	Yes
expandProgram OR relationship	Same as for a structure browser: the relationship or expandProgram used to expand the objects being compared	Yes
level	The number of levels to expand the structure before initiating the comparison processing. The level can be: Passed as the level URL Parameter Posted as an HTML form element with the element name level. Default=1.	No. If not passed, the structure is expanded by 1 level.
matchBasedOn	1 or more columns used to determine if children of Object 1 and Object 2 are matches. The matched column names can be: Passed as a comma-separated list in the URL Parameter matchBasedOn Posted as an HTML form element with the element name MatchBasedOn. Each form element must have this name and contain the unique values for that specific match.	Yes
objectId	The objectIds of the objects being compared. These ids can be: Passed as a comma-separated list in the URL Parameter objectId (2 id's must be passed) Posted as an HTML form element with the element name objectId.	Yes
reportType	The type of report to generate. The report type can be: Passed as the reportType URL Parameter Posted as an HTML form element with the element name reportType. The possible values for the URL Parameter or form element are: complete_summary (default) difference_only Unique_toLeft_Report Unique_toRight_Report Common_Report	No. If not passed, complete_summary is used.
table	The system table name to use in the compare report page. The table can be: Passed as the URL Parameter table Posted as an HTML form element with the element name table.	Yes

In general, you follow these steps to create a custom structure compare dialog:

1. Clone the AEFStructureCompare command.
2. Edit the href setting to call a custom JSP page instead of emxStructureCompare.jsp.
3. Connect the cloned command to the needed toolbar. Make sure that the toolbar does not also have the AEFStructureCompare command attached to it.

Structure Compare Report

The emxStructureCompareReport.jsp can produce 5 types of reports. A complete summary or a Difference Only Summary.

The emxStructureCompare.jsp or custom JSP passes in 1, 2, or 3 matching criteria column names, and any number of comparison column names. The report sorts the structures based on these fields, and then matches and compares the child objects. The reports do not compare the root nodes for the two objects.

This section discusses the following topics:

- [Complete Summary Report](#)
- [Difference-Only Report](#)
- [Unique to Left and Unique to Right Reports](#)
- [Common Components Report](#)

Complete Summary Report

When the user selects Complete Summary in the default Structure Compare dialog box, or a custom JSP passes the `reportType=complete_summary` URL parameter, the report lists all child objects of Object 1 and Object 2 with visual indicators of the difference between the child objects.

Comparison Report : Complete Summary Report											
Acme Widget And Amalgamated Broom											
Sync To Left		Sync To Right									
Name	Type	Revision	Description	Name	Type	Revision	Description	Cust ID	Street A	City	
Acme Widget	AEFQE	1	A Widget	Amalgamated Broom	AEFQE	S-1	Brooms Unite!	AB	2525 Sweep Street	PU	
-> Payment AW-002	AEFQE	1	Decor for	Payment AB-001	AEFQE	S-1	Decor for Payment AB-001				
-> Payment AW-004	AEFQE	1	Decor for	Payment AB-003	AEFQE	S-1	Decor for Payment AB-003				
-> Payment AW-005	AEFQE	1	Decor for	Payment AB-005	AEFQE	S-1	Decor for Payment AB-005				
-> Payment AW-003	AEFQE	1	Decor for	Payment AB-007	AEFQE	S-1	Decor for Payment AB-007				
-> Payment AW-001	AEFQE	1	Decor for	Payment AB-009	AEFQE	S-1	Decor for Payment AB-009				
->				Payment AB-011	AEFQE	S-1	Decor for Payment AB-011				

The report lists all columns in the table except those defined with the `Column Type` is `Icon`, `Image`, or `Separator`. To explicitly omit a column from this report, define the `Compare Report=false` setting for that column. The column headings shown in red are the ones selected for comparison.

Matched objects (child objects of Object 1 and Object 2 have the same values for all specified Match Based On columns), display side by side on the page. If a child object of either Object 1 or Object 2 does not match any child objects of the other, it only appears on one side of the page and is highlighted in yellow. Child objects of these unique objects are not compared.

For matched objects, a column value highlighted in blue indicates a value that differs between Object 1 and Object 2.

The Diff Code column indicates how the 2 objects differ for that row. The Structure Compare feature comes with these default code values:

- ADD. Indicates the object is unique to the left side of the report.
- DEL. Indicates the object is unique to the right side of the report.

You can provide code values on columns to show in the Diff Code column when the values in those columns differ between objects 1 and 2. The syntax for this setting is:

```
Diff Code = <precedence>:code value>
```

The precedence must be an integer and it defines which code value to use when the objects have multiple columns with different values. The code value can indicate the column name, or the type of data in the columns, or any needed value.

Multiple columns can have the same precedence value. As an example, the settings on 2 columns could be:

```
Diff Code = 1:Attribute
```

```
Diff Code = 2:Class
```

If both of these columns differ between the 2 objects, the Diff Code column shows `Attribute`, because that has a lower precedence. If the columns that differ have the same precedence, both code values show in the Diff Code column.

The Complete Summary Report includes the standard AEFStructureCompareToolbar. You can customize the page by passing the `toolbar=menuname` URL Parameter and use a custom toolbar. See "Comparing the Structure of Objects" in the *Application Exchange Framework User's Guide* for description of the results page and how the user can synchronize the structures to the right or to the left.

If the synchronize process adds a new row to an object, the connection is based on the relationship used in the source structure for the node. If the relationship is not valid or the user does not have connect access, the change is not made and the user is notified. To use custom logic for making this connection, you can pass the `connectionProgram` URL parameter that defines the custom JSP to perform this task.



Difference-Only Report

When the user selects Difference Only Summary in the default Structure Compare dialog box, or a custom JSP passes the `reportType=difference_only` URL parameter, the report only lists those objects that only appear in one of the Object's structures, or have different values in the columns selected for comparison. Any child objects that are the same for Object 1 and Object 2 do not show in this version of the report. In addition, the sync functions are not available for this report.

Comparison Report : Difference Only Report								
Level	Name	Type	Revision	Name	Type	Revision	Description	Cust ID
0	Acme Widget	AEFQE_	1	Amalgamated Broom	AEFQE_S	1	Brooms Unite!	45
1				Invoice AB-001	AEFQE_S	1	Descr for Invoice AB-001	
1				Invoice AB-002	AEFQE_S	1	Descr for Invoice AB-002	
1				Invoice AB-003	AEFQE_S	1	Descr for Invoice AB-002 Descr for Invoice AB-003	

27 objects

As with the Complete Summary Report, if a child object of either Object 1 or Object 2 does not match any child objects of the other, it only appears on one side of the page and is highlighted in yellow. Child objects of these unique objects are not compared.

For matched objects, a column value highlighted in blue indicates a value that differs between Object 1 and Object 2.



Unique to Left and Unique to Right Reports

When the user selects Unique to Left or Unique to Right in the default Structure Compare dialog box, or a custom JSP passes the `reportType=Unique_toLeft_Report` or `Unique_toRight_Report` URL parameter, the report only lists those objects that only appear as child objects of Object 1 (Unique to Left) or Object 2 (Unique to Right) in the columns selected for comparison. Left and right are as shown on the Complete Summary Report; that is, Object 1 shows on the left and Object 2 shows on the right. Any child objects that are the same for Object 1 and Object 2 do not show in these versions of the report. The sync functions are not available for this report.

Comparison Report : Unique to Left Report

Acme Widget And Amalgamated Broom

Level	Name	Type	Revision	Description	Cust ID	Street A	City
0	Acme Widget	AEFQE_S	1	A Widget a Day Keeps the IRS Away	12	167 Widget Way	
1	Invoice AW-001	AEFQE_S	1	testqwe			
1	Invoice AW-002	AEFQE_S	1	Descr for Invoice AW-002 werwe werre			
1	Invoice AW-003	AEFQE_S	1	Descr for Invoice AW-003			
1	Invoice AW-004	AEFQE_S	1	Descr for Invoice AW-004			

6 objects

This example shows the Unique to Left Report, showing child objects that are unique to Object 1 (that shows on the left side of the complete summary report).

Common Components Report

When the user selects Common Components Report in the default Structure Compare dialog box, or a custom JSP passes the `reportType=Common_Report` URL parameter, the report only lists those child objects with matching values in the columns selected for comparison. The sync functions are not available for this report.

Comparison Report : Common Components Report - Mozilla Firefox

Acme Widget And Amalgamated Broom

Level	Name	Type	Revision	Name	Type	Revision	Description	Cust
0	Acme Widget	AEFQE_S	1	Amalgamated Broom	AEFQE_S	1	Brooms Unite!	4

1 object

Done

In the above example, there are no matching child objects for the selected attributes.

URL Parameters Accepted by emxStructureCompare.jsp and emxStructureCompareReport.jsp

If the needed URL parameter was passed to emxIndentedTable.jsp (to the page displaying the structure browser from which the compare is initiated), you do not need to explicitly pass them to emxStructureCompare.jsp and emxStructureCompareReport.jsp.

URL Parameter	Description	Accepted Input Value
calculations	When true, the calculations toolbar icon (Σ) displays on the structure browser toolbar; when false, the icon does not display.	true (default) false
compareBy	One or more column names from the table. The value can be a comma-separated list or posted as an HTML form element with the name "compareBy", with each column name its own form element with a unique value.	Quantity,Find Number
connectionProgram	Defines the JPO:method to connect objects in a given structure hierarchy. You only need to write and specify a connectionProgram if you need to implement custom logic for connecting objects.	emxPart:connectEBOMParts emxDocument:connectFolders
expandProgram	Define the JPO:method to use for expanding the object.	emxPart:getEBOM emxDocument:getFolders
level	The number of hierarchical levels to traverse to fetch object details. The level must be an integer value.	1 (default)
matchBasedOn	One or more column names from the table. The value can be a comma-separated list or posted as an HTML form element with the name "matchBasedOn", with each column name its own form element with a unique value.	Type,Revision
objectId	The Object IDs of the objects to be compared. The value can be a comma-separated list or posted as an HTML form element with the name "objectId".	
preProcessJavaScript	Defines a JavaScript function and is called after the structure browser is loaded in edit mode to execute custom processing. Can be a single function, or a semicolon-separated list of functions.	JavaScript function name
relationship	<p>Relationship names used to expand the root object to display the table structure view. A comma-separated list of relationships can be defined.</p> <p>The same relationships are used to expand the structure by clicking on the plus on the nodes.</p> <p>The relationship name passed in can be the symbolic name or the actual relationship name. Using the symbolic names is recommended.</p> <p>When one or more relationships are passed in, the program uses the list of passed-in relationship(s) only to expand the object. If no relationship parameter is passed in, then it assumes all.</p> <p><all> (default) - The given object will be</p>	relationship_EBOM, relationship_PartSpecification all (default)

	expanded to get all the connected objects, irrespective of what relationship is used, to be displayed as child/parent objects.	
reportType	The type of report to generate. See Structure Compare Report	complete_summary (default) difference_only Unique_toLeft_Report Unique_toRight_Report Common_Report
table	Passes the table definition for displaying the criteria dialog and the compare report.	ENCBOMSummary PMCAIIProjects
toolbar	Defines the name of the toolbar menu to display as a toolbar in the compare report. By default, the AEFStructureCompareToolbar toolbar with synchronize buttons is used.	AEFStrutureCompare ENCEBOMToolbar PMCFolderToolbar

Streaming Query and Expand

Enabling streaming through the classes usually provides the greatest improvement in performance and memory consumption.

In this section:

- [About Streaming Query and Expand](#)
- [Enabling Streaming Query in a Custom Application](#)
- [Enabling Streaming Expand in a Custom Application](#)

About Streaming Query and Expand

You can enable streaming in a query or an expand operation in order to fetch the results one page at a time. When you enable streaming, data is returned when the first page is available, resulting in improvement in performance and memory consumption. When streaming is used, no data is held in cache. Therefore repeating the same operation in a transaction can result in the system issuing the same sql database requests each time.

Streaming query and expand is enabled by default. To disable this function, edit the enovia.ini file with the following setting:

`MX_QUERY_PAGE_SIZE=0`

Only results from local vaults are streamed.

Streaming can be enabled through the MX_QUERY_PAGE_SIZE environment variable, through classes in the Studio Customization Toolkit, or at runtime by using the size clause in MQL temporary query or expand business object commands. You can override any value set for streaming in temporary queries and expands by using the size clause. For information on MX_QUERY_PAGE_SIZE, see "Optional Variables" in the *ENOVIA Live Collaboration Installation Guide*. For information on using the size clause, see "Reference Commands" in the *MQL Guide*.

Enabling streaming through the classes usually provides the greatest improvement in performance and memory consumption. The following iterator classes are used: QueryIterator, RelationshipQueryIterator and ExpansionIterator. These classes must be defined, used, and closed within a single transaction boundary. The transaction must be started before using getIterator and the iterator must finish its work before the transaction is completed. If these classes are not used within a single transaction, the following error will occur: `QueryIterator must be used within a transaction`.

Enabling Streaming Query in a Custom Application

You can enable streaming query in a custom application.

1. Open `Query.java`.

2. Replace the following section:

```
BusinessObjectWithSelectList list = query.evaluateSelect(cntx,  
selects);  
  
For (int i=0; i<list.size(); i++) {  
    BusinessObjectWithSelect bo = list.getElement(i);  
?  
}  
  
With this:
```

```
QueryIterator itr = query.getIterator(cntx, selects, pageSize);  
while (itr.hasNext()) {  
    BusinessObjectWithSelect bo = itr.next();  
?  
}
```

Enabling Streaming Expand in a Custom Application

You can enable a streaming expand in a custom application.

1. Open BusinessObject.java.

2. Replace the following section:

```
Expansion expansion = bo.expandSelect(cntx, relPattern,
typePattern,busSelects, relSelects, getTo, getFrom, levels,
limit, checkHidden);

BusinessObjectWithSelect root = expand.getRootWithSelect();
RelationshipWithSelectList rels = expand.getRelationship();
for (int i=0; i<rels.size(); i++) {
RelationshipWithSelect rel = rels.elementAt(i);
?
}
```

With this:

```
ExpansionIterator itr = bo.getExpansionIterator(cntx,
relPattern, typePattern,busSelects, relSelects, getTo, getFrom,
levels, limit, checkHidden, preventDups, pageSize);

BusinessObjectWithSelect root = itr.getRoot();
while (itr.hasNext()) {
RelationshipWithSelect rel = itr.next();
?
}
```

Additional URL Parameters for Structure Browsers

In addition to directly defining the URL parameters to pass to emxIndentedTable.jsp, you can use the `appendURL` parameter to retrieve additional URL parameters from the properties file for the ENOVIA application.

In addition to directly defining the URL parameters to pass to emxIndentedTable.jsp, you can use the `appendURL` parameter to retrieve additional URL parameters from the properties file for the ENOVIA application. The `appendURL` parameter takes a value in this format:

`<KEY>|SuiteKey`

Where the `KEY` is a component of a property in the properties file of the specified `SuiteKey`. For example,
`appendURL=Effectivity|EngineeringCentral`

The system uses this value, plus the UI component where the `appendURL` is defined (in this case a structure browser), to fetch the list of additional URL parameters to pass to the UI component. The above example corresponds to this property in the `emxEngineeringCentral.properties` file:

```
emxEngineeringCentral.StructureBrowser.Effectivity =  
appendColumns=AdditionalColumnsTable&toolbar={$ORIGINAL},effectivitytoolbar
```

In this example, 2 URL parameters are being returned to the Structure Browser page that specified the `appendURL` parameter: `appendColumns` and `toolbar`. The `$_ORIGINAL` macro is used to replace the URL parameter in the original URL string with the one specified in this property. In this example, the toolbar defined in the URL string would be replaced by the one named `effectivitytoolbar`.

As shown by the above example, the `emxSUITENAME.properties` file must contain the required key specified in this format:

`emx<SuiteKey>.<ComponentKey>.<KEY>=VALUE`

where:

- `emx<SuiteKey>` is the application's suite key, such as `emxEngineeringCentral`
- `<ComponentKey>` is the type of UI component requesting the additional URL parameters and is one of these values:
 - Create
 - Form
 - Table
 - StructureBrowser
- `<KEY>` is the specific key being looked up
- `VALUE` is an ampersand-separated list of URL parameters

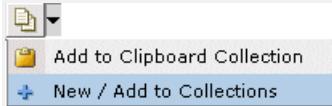
URL Parameters Accepted by emxIndentedTable.jsp

This table lists the parameters that you can use with emxIndentedTable.jsp. You can add these parameters to the href parameter for the component that calls the structure browser. They provide the main configurability control to the structure browser component.

URL Parameter	Description	Possible / Default Values
addJPO	The JPO name and method name to execute when the user adds an empty row to the structure browser for creating a new object. If both the <code>addJPO</code> and <code>applyURL</code> parameters are defined, the <code>addJPO</code> URL parameter is ignored.	JPOName:methodName
appendColumns	The name of the table that defines columns that you want to include in the structure browser being defined.	
applyURL	Specifies the URL string to be submitted when a user clicks the Apply button after making changes to the structure browser in edit mode. If both the <code>addJPO</code> and <code>applyURL</code> parameters are defined, the <code>addJPO</code> URL parameter is ignored.	
autoFilter	Determines whether the filter tool  displays in the toolbar, and overrides the Auto Filter setting on a table or structure browser column.	true (default)--The Filter tool displays in the page toolbar. false--The Filter tool does not display in the page toolbar.
calculations	Controls the display of the Σ toolbar button. When true, this icon only displays on the toolbar if the structure browser contains at least one column defined as numeric.	true--Sigma icon shows if the structure browser contains at least 1 numeric column false (default)--Sigma icon does not show on the toolbar.
cancelLabel	Used only when the structure browser is in a pop-up window and in View mode. It is ignored for all other conditions. Shows a Cancel button in the footer that closes the pop-up window without any other actions. Can be a text string or a string resource id.	Cancel emxFramework.FreezePane.Close
connectionProgram	The JPO:method used for operations such as resequence, Copy, Paste, and Cut & Paste to make the needed connection between child and parent objects. This parameter is only used to implement custom logic.	emxPart:connectEBOMParts emxDocument:connectFolders
customize	Enables (true) or disables (false) the ability for users to create customized versions of the page. This parameter overrides the <code>emxFramework.UITable.Customization</code> system property, and has no effect on other table or structure browser pages. If not provided, the value specified for <code>emxFramework.UITable.Customization</code> is used. All access controls defined for the system table are retained in the customized table. If a column does not have a label or alt value defined (and is not a file, checkbox, icon, image, or separator), the column name shows in the user's selection list, however, this name cannot be internationalized.	true false
direction	Direction use for expanding the object. When one or more relationship names are passed in, the object is expanded for the given relationship(s) using the specified direction. Not supported for non-object based structure browsers.	to from both (default)
directionFilter	Used to turn ON or OFF the direction filter shown in the header. By default the parameter is false and the direction filter is hidden. The parameter must be passed explicitly as true to show the filter. Not supported for non-object based structure browsers.	true false (default)
editLink	Specify whether the view table should display the Mode button toggle button. When true, the Mode menu and disabled Edit menu (AEFSBEditActions) is included in the toolbar.	true false (default)
editRelationship	Defines a comma-separated list of relationships that can be used in conjunction with the cut and paste edit commands. The user can only cut or paste rows that are connected to the parent object using one of the defined relationships.	relationship_AffectedItems, relationship_EBOM
expandLevelFilter	Enable or disable the Expand filter for the structure browser page. This parameter overrides the <code>expandLevelFilterMenu</code> URL parameter.	true (default) false

expandLevelFilterMenu	Specify the menu used to show the Expand level filter.	Administrative menu name AEFFreezePaneExpandLevelFilter (default)
expandProgram	Used to pass the name of the program to use for expanding the object. This parameter value should be assigned to the JPO name and the method name separated by a colon.	<JPO Name:method Name> emxPart:getEBOM emxDocument:getFolders
expandProgramMenu	<p>Can be assigned to an administrative menu or a command name, which contains the definitions of the JPO, method name and label.</p> <p>When command is passed in: Command settings <code>program</code> and <code>function</code> are used for getting the JPO name and method name for expanding the objects.</p> <p>The List Filter is not shown because there is only one JPO available to expand.</p> <p>When Menu is passed in: Commands connected to this menu are used for listing the options in the List Filter combo box. The system obtains the label for each item from the command's label.</p> <p>The <code>program</code> and <code>function</code> settings on individual commands are used as the JPO method for expanding the objects, upon selecting the options from the List Filter.</p> <p>By default, the first command in the list is used for expanding the objects.</p> <p>The commands connected to the menu honor all the access settings supported by the configurable component (such as Access Mask, Access Expression, and Access Program).</p>	Administrative menu name Administrative command name For example: ENCBOMLists (menu) PMCFolderLists(menu) ENCBOMList (command)
Export (view mode only)	<p>Shows or hides the Export  tool on the page toolbar. When users click the tool, the system exports the table data to a file in the user's preferred format in their preferred format: CSV, HTML, or Text. Users choose their preferred format using the Preferences tool in the global toolbar. For information on configuring the export preference, see About Preferences. For instructions on using the preference page, click Help from the preference page.</p> <p>If the user chooses CSV or Text for their export format, the export does not include Icon column type and programHTMLOutput columns.</p> <p>To get the column values exported when the column type is "programHTMLOutput," set "Export= true" for the column.</p>	true (Default)--The Export tool displays, allowing users to export the table data. false--The tool does not display.
findMxLink	When true, show the mxLink icon/command button on the toolbar, which opens a search dialog box.	true (default) false
freezePane	Used to configure which column(s) should display as the freeze pane column in the structure browser. If this parameter is not passed in, the first column in the table is used as Freeze Pane column. A comma-separated list can be provided, and the columns are displayed in the order listed. If all of the table columns are passed, the last column listed for this parameter displays in the scroll pane.	Column name in the table Name Title Name,Title,Description
header	The content of the heading that appears at the top of the table page.	Any alphanumeric text or a string resource ID. header=Buyer Desk header=emxQuoteCentral.AssignedPackages.AssignedPackages
HelpMarker	Specifies the name of the help marker to call for context-sensitive help. For information about implementing help, see Implementing Context-Sensitive Help for FrameMaker Source .	String The naming convention for help markers "emxhelp" followed by the object or feature and then the action, for example, emxhelprouutecreate and emxhelpprojectedit. The marker is all lowercase with no spaces.
hideRootSelection	Shows or hides the check box for the root node when the structure browser is configured for a single root node and the <code>selection</code> URL parameter is set to multiple (ignored if selection is set to single or none or the structure browser includes multiple root nodes).	true false (default)
jpoAppServerParamList	<p>Allows session data to be passed to a JPO and uses the format:</p> <pre>scope:attributeName</pre> <p>where scope can be one of these values:</p> <pre>application session request</pre> <p>and the attribute must be a valid attribute used within the specified scope. The parameter can pass a comma-separated list of scope:attributeName values. The</p>	application:<attributeName>,session:<attributeName>,request:<attributeName>

	attribute values must be serializable.	
lookupJPO	The JPO name and method name to execute the lookup function for a row added to the structure browser (using +). The JPO is invoked when the user clicks Lookup Entry .	JPOName:methodName
massPromoteDemote	Overrides the system property <code>emxFramework.Lifecycle.MassPromoteDemote.Enable</code> for the specific page. The default value for this parameter is the value for the above property. The page must also include the State and Type columns. See the <i>Live Collaboration Administrator's Guide</i> for details on setting property values that control the mass promote/demote feature.	true false
massUpdate (edit mode only)	Use to turn on or off mass update controls on the editable table page.	true (default)--Mass update is available for the page. false--Mass update not is available for the page.
mode	Indicates what mode to display the structure browser. This parameter is optional.	edit view (default)
multiColumnSort	Enables or disables multiple column sorting on a page. If disabled, the user can still sort by a single column if one is specified in the sortColumnName parameter. When set to false, multiple column sorting is also disabled in any custom tables users' create based on this system table.	true (default) false
objectId	Required parameter for this component. It represents the object ID of the root object in the structure. For multiple root nodes, the <code>program</code> parameter is used instead, however, if the program is not available, this id is used to get the root object details.	<OID for the root object in the structure>
onReset	The name of a JavaScript function that is invoked when a user clicks the Reset button on an editable Structure Browser page. The function must be included in the file defined in <code>emxSystem.properties: eServiceSuiteAPPLICATIONNAME.UIFreezePane.ValidationFile = VALIDATIONFILENAME</code> (either .js or .jsp) For example: <code>eServiceSuiteEngineeringCentral.UIFreezePane.ValidationFile = emxEngineeringCentralFormValidation.jsp</code>	resetEverything
PrinterFriendly (view mode only)	Specifies whether the table page should include the Printer Friendly tool. You can have the system pass this parameter automatically by entering the Printer Friendly setting for the command object that calls the table page.	true (default) false
program	Specifies the JPO and method to generate the list of toot objects. When this parameter is passed, the <code>objectId</code> URL parameter is ignored. The JPO returns one or more objects.	<JPO name:Method name> AEFSearch:getPartList
relationship	Relationship names to be used for expanding the root object to display the table structure view. The same relationships are used while expanding the structure by clicking on the plus on the nodes. The relationship name passed in can be the symbolic name or the actual relationship name. Using the symbolic names is recommended. A property setting can be used to pass the relationship names. The property must be assigned to one or more relationships to be used for expansion. When one or more relationships are passed in, the program uses the list of passed-in relationship(s) only, to expand the object. If no relationship parameter is passed in, then it assumes all. <all> (default) - The given object will be expanded to get all the connected objects, irrespective of what relationship is used, to be displayed as child/parent objects. Not supported for non-object based structure browsers.	<relationship name> <list of comma separated relationship names> <property key assigned to one or more relationships> For example: relationship_EBOM, relationship_PartSpecification relationship_Employee all (default)
relationshipFilter	Used to turn ON or OFF the relationship filter shown in the header. By default the parameter is false and the relationship filter is hidden. You must explicitly pass the parameter as true to show the filter. Not supported for non-object based structure browsers.	true false (default)
resequenceRelationship	Used when the order of child objects is significant. Defines a comma-separated list of relationships that determine if a paste operation is a resequencing. Resequencing means that the child object was moved from one place in the list of child objects to another (under the same parent object).	relationship_AffectedItems, relationship_EBOM

selection (view mode only)	<p>Controls whether the table page adds a column of check boxes or radio buttons in the left-most column of the table. When set to <code>single</code> or <code>none</code>, the parameter <code>objectCompare</code> is false and the object compare icon does not display.</p>	<p>multiple--Users can select more than one row in the table. A check box displays in the left column of each row. There are no access restriction for the check boxes. If you pass this parameter, it overrides any check box column added to the table administrative object. For more information, see Column Values as Check Boxes.</p> <p>single--Users can select one row in the table. A radio button displays in the left column of each row.</p> <p>none--A selection column is not added to the table page. (A check box column can still be displayed by adding the column to the table administrative object.)</p>
showApply	When set to true, shows the Apply button in Edit mode. When set to false, Edit mode does not have an Apply button.	true (default) false
showClipboard	<p>This menu is enabled by default and adds the  to the page toolbar. The icon acts as a pull-down menu:</p>  <p>If the user clicks , selected objects are added to the clipboard collection for that user. If the user clicks the arrow, the user can select the New/Add to Collections or Add to Clipboard Collection command.</p> <p>Set the value for this parameter to false to disable this feature.</p> <p>For File Summary Pages for ENOVIA products, the default value for this parameter is false.</p>	true (default) false
showPageURLIcon	<p>When true,  shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false,  does not show in the toolbar.</p> <p>The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in <code>emxSystem.properties</code>.</p>	true false
showRMB	Enables or disables the right-click menus on the page. When set to false, all right-click menus for that page are disabled.	true (default) false
showTabHeader	<p>Applies only in Portal mode (when the page is displayed within a PowerView), enables or disables the page header.</p> <p>If false, the header text is not displayed in the PowerView tab.</p> <p>If true, the header text shows in the tab.</p>	true false (default)
sortColumnName	<p>The column names to use for sorting the column data, or <code>none</code> if no initial sorting is required. When the node is expanded, the system uses the same parameter to sort the list and display the child objects. By default, the first columns in the table view (Freeze Pane column) are used to sort. The sorting is "alphanumeric" (<code>sortType=alpha</code>).</p> <p>Can be a comma-separated list of up to 3 column names. The page sorts by the first provided column, then by the second, then by the third.</p> <p>Used as the default Sort by settings in the Customize Table View dialog box.</p> <p>Columns defined with <code>sortType=other</code> setting cannot be used with <code>multiColumnSort</code>.</p>	<name of the column> column1,column2,column3 none
sortDirection	<p>Defines the sort order for the columns specified in the <code>sortColumnName</code> parameter. If a single value is passed, it applies to all columns, or you can pass a comma-separated list of values matching the number of columns in the <code>sortColumnName</code> parameter.</p> <p>Used as the default sort directions in the Customize Table View dialog box.</p>	ascending (default) descending ascending,descending,ascending
subHeader	Creates a subHeader below the main header in the table header frame.	<p>The value can be any static text or a string resource id. For example: <code>subHeader= emxEngineeringCentral.Common.BOMLevel</code> <code>subHeader=Bill of Material Level 1</code></p> <p>The value can also include macros such as <code>\$<type> \$<revision></code>.</p>
submitLabel	For View mode only, shows a Submit button in the footer. If this URL parameter is not passed and <code>SubmitURL</code> is passed, then the default "Submit" is used as the label. Can be a text string or a string resource id.	Submit <code>emxFramework.Common.Submit</code>

SubmitURL	The JSP called when a user clicks the Submit button in the structure browser footer.	emxProcess.jsp \${SUITE_DIR}/emxProcess.jsp
table	Passes the table definition for displaying the structured view.	<table name> - system table For example: ENCBOMSummary PMCAIIProjects
tableMenu	<p>Can be assigned to an administrative menu or command name that contains the table name definition and label.</p> <p>When you pass command:</p> <p>Command setting <code>table</code> gets the table name for the table definition.</p> <p>The Table Filter does not display (because there is only one table available to view).</p> <p>When you pass menu:</p> <p>The system uses commands connected to this menu for listing the options in the Table Filter combo box. The system obtains the label for each item from the command's label.</p> <p>The system uses the <code>table</code> setting on individual commands as the table definition for displaying the view, upon selecting the options from the Table Filter.</p> <p>By default, the system uses the first command for the table definition.</p> <p>The commands connected to the menu honor all the access settings supported by the configurable component (such as Access Mask, Access Expression, and Access Program).</p>	Administrative menu name Administrative command name For example: ENCBOMViews (menu) PMCFolderViews(menu) ENCBOMView (command)
TipPage (view mode only)	Specifies whether the table page should include the Tip Page tool and call a specific html or jsp when a user clicks the tool. If this setting is not included, the Tip tool is not included on the table page.	Name of a custom html or JSP page, including any path. The starting point for the directory reference is the content directory. For example, if you want to call an html file in ematrix/doc/customcentral and the content directory is ematrix/customcentral, you would add this parameter to the table.jsp: TipPage=../doc/custom-tailored/tippage.html
toolbar	A comma-separated list of UI menu names that add custom filters to the toolbar. Menus specified here are added as a new toolbar row beneath the Action toolbar.	PMCWBSTaskToolbar ECEBOMToolbar,ECEBOMFilter
TransactionType	Controls whether the table query is run within an Update transaction or Read transaction. Use Update transactions whenever the system need to update the database while fetching the object list.	read (default)--The table query is not run within an Update transaction. update--The table query (inquiry or JPO) runs within an Update transaction.
triggerValidation	Controls the Validate command on the toolbar. If true or empty, the command shows on the toolbar. If set to false, the trigger validation icon does not display on the toolbar.	true (default) false
type	<p>Used to pass the type name for filtering the expanded structure when the table displays and also when the structure expands.</p> <p>The type name can be the symbolic name or the actual type name. Using the symbolic name is recommended.</p> <p>You can use a property setting to pass the type names. The property must be assigned to one or more types for filtering.</p> <p>If you pass one or more types, the program uses the list of passed-in types to filter the object list. If you pass no parameter, the system uses all.</p> <p>all (default) - Displays all objects and performs no filtering.</p> <p>This is an optional parameter.</p> <p>Not supported for non-object based structure browsers.</p>	<type name> <list of comma separated type names> <property key assigned to one or more types> For example: type_Part,type_ECO type_Folder all (default)
typeFilter	Used to turn ON or OFF the type filter shown in the header. By default the parameter is false and the type filter is hidden. You must explicitly pass the parameter as true to show the filter.	true false (default)
view	<p>A comma-separated list of available views for the structured data. This parameter supports details and thumbnails views.</p> <p>This URL parameter override the emxFramework.Freezepane.view property in emxSystem.properties.</p>	details details,thumbnails

Settings for Columns in a Structure Browser

This table lists and describes the related column settings you can use to control the structure browser component behavior. The name and value for each setting is case sensitive.

Most of the settings supported by configurable tables are supported by the structure browser.

Setting	Description	Accepted Values/Examples
Access Expression	Controls access to table columns just as they can be used to control access to other UI components. For details, see User Access to UI Components .	--
Access Function	Access Expression and Access Mask are not supported for non-object based structure browsers.	
Access Mask	If the displayMode setting is also used on the column, these settings should not be used.	
Access Program		
Admin Type	Use to translate columns whose values are administrative types. The value is translated and presented. For attributes, provide the symbolic name of the attribute. Symbolic names start with "attribute_" and are followed by the attribute name with no spaces. Not supported for non-object based structure browsers.	Type State Role Relationship attribute_UnitOfMeasure
Allow Manual Edit	When true, users can manually edit the table column. Applicable only when you set the range parameter to a URL or when you assign the setting format to date or for fields of type <code>combobox</code> . The system ignores all other cases. When true, the system ignores the Admin Type setting. Not supported for non-object based structure browsers.	false (default)--Manual entry is not allowed. true--Manual entry is allowed.
Alternate OID expression	By default, when a column's data is configured to show as a hyperlink using the href parameter, the system passes the ID of the object for that row in the objectId parameter. Using this setting, you can configure the hyperlink so a different objectId is passed. The system passes the ID for the object returned from the expression defined in this setting. Not supported for non-object based structure browsers.	\$<to[relationship_NewPartPartRevision].from.id> \$<to[relationship_EBOM].from.id> For more information on configuring a column using an alternate object ID's expression, see Column Values Using Alternate OID in href and Select Expression and Column Values Using Alternate OID and Type Icon in href with Select Expression .
Alternate Policy expression	This setting is required to display the state of any connected objects and show the value translated. This setting is applicable only when the Admin Type setting is set to State. Not supported for non-object based structure browsers.	Any select expression that evaluates to a policy of a connected object.
Alternate Type expression	When the Show Alternate Icon setting is true, this expression is used to obtain the object type. Based on the obtained type, the corresponding icon is displayed. For more information on configuring a column using an alternate type expression, see Column Values Using Alternate OID in href and Select Expression . Not supported for non-object based structure browsers.	\$<to[relationship_NewPartPartRevision].from.type> \$<to[relationship_EBOM].from.type>
Arithmetic Expression	Comma-separated list of arithmetic expressions. Expressions can include cell variables.	
Auto Filter	Determines whether users can filter the structure browser rows based on data in the column. If at least one column in the table has Auto Filter set to true, the Filter  tool displays in the page toolbar. If passed, the autoFilter URL Parameter overrides this	true (default)--The Filter tool displays in the page toolbar. When clicked, the column is available on the Auto Filter Selection page. false--The column is not available in the Auto Filter Selection page.

	<p>setting.</p> <p>Filtering applies to root level objects only.</p> <p>See Filter for a Structure Browser Page.</p>	
Calculate Average	When true, calculates the average of values in the column, using the defined decimal precision. By default, the calculation uses the system-wide precision (set in emxSystem.properties), or you can use the Decimal Precision setting to set a specific precision for this column calculation.	true false (default)
Calculate Average Label	When the Calculate Average setting is used, any value for this setting shows as the label for the column average. If not defined, "Average" is used as the label.	Average (default) <any string value>
Calculate Custom	When true, invokes the program defined by the Calculate Custom Program setting.	true false (default)
Calculate Custom Label	When the Calculate Custom and Calculate Custom Program settings are used, any value for this setting shows as the label for that calculated value. If not defined, "Custom Label" is used as the label.	Custom Label (default) <any string value>
Calculate Custom Program	If the Calculate Custom setting is true, this setting defines the program and function to invoke. The program can perform any calculations needed for the data in the column, with a single result.	program:function
Calculate Maximum	When true, displays the largest value of the column values.	true false (default)
Calculate Maximum Label	When the Calculate Maximum setting is used, any value for this setting shows as the label for the column maximum. If not defined, "Maximum" is used as the label.	Maximum (default) <any string value>
Calculate Median	When true, displays the middle value (the mean) of the column values. If the column contains an even number of values, the average of the 2 middle numbers is used, using the defined Decimal Precision. By default, the calculation uses the system-wide precision (set in emxSystem.properties), or you can use the Decimal Precision setting to set a specific precision for this column calculation.	true false (default)
Calculate Median Label	When the Calculate Median setting is used, any value for this setting shows as the label for the column median. If not defined, "Median" is used as the label.	Median (default) <any string value>
Calculate Minimum	When true, displays the smallest value of the column values.	true false (default)
Calculate Minimum Label	When the Calculate Minimum setting is used, any value for this setting shows as the label for the column minimum. If not defined, "Minimum" is used as the label.	Minimum (default) <any string value>
Calculate Standard Deviation	<p>When true, displays the standard deviation of the column values using this formula:</p> $sd = \sqrt{\frac{\sum (x - \bar{x})^2}{N-1}}$ <p>where:</p> <p>N=total number of elements</p> <p>xbar=mean of the column values</p> <p>By default, the calculation uses the system-wide decimal precision (set in emxSystem.properties), or you can use the Decimal Precision setting to set a specific precision for this column calculation.</p>	true false (default)

Calculate Standard Deviation Label	When the Calculate Standard Deviation setting is used, any value for this setting shows as the label for the column calculated value. If not defined, "Standard Deviation" is used as the label.	Standard Deviation (default) <any string value>
Calculate Sum	When true, displays the total of the column's values.	true false (default)
Calculate Sum Label	When the Calculate Sum setting is used, any value for this setting shows as the label for the column sum. If not defined, "Total" is used as the label.	Total (default) <any string value>
Calendar Function	Use to specify the name of the method in the JPO specified in the Calendar Program setting that retrieves the non-working days based on the calendar defined for the location.	The name of a function in the Calendar Program JPO, such as: <code>getNonWorkingDays</code>
Calendar Program	Use to specify the name of a JPO that contains a method to get the non-working days for a calendar.	The name of a JPO, such as: <code>emxWorkCalendar</code>
Column Icon	Use to display an icon for the column's data instead of other data. Required when the setting "Column Type" is assigned to "icon". Also see Column Values as Icons . Not supported for non-object based structure browsers.	Name of an image file such as: <code>images/NewWindow.gif</code> <code>images/EditItem.gif</code>
Column Type	<p>The setting "Column Type" is used when no expression is defined for the column data.</p> <p>For recommendations on improving the performance of table columns whose data is generated with a JPO program (Column Type set to program or programHTMLOutput), see Improving Performance of Table Columns.</p> <p>If using programHTMLOutput to populate a column with an image or icon, make sure to include a height attribute to prevent column/row misalignments, for example:</p> <pre></pre> <p>For non-object based structure browsers, only these column types can be used:</p> <ul style="list-style-type: none"> program programHTMLOutput separator 	<p>program--The values for this column are obtained from a program (JPO). With this setting, the program and function name are required as settings. For more information on configuring a column using a program, see Column Values as Select Expressions.</p> <p>programHTMLOutput--Same as the "program" setting above, except the column value output is in HTML format. Column values are placed in the table cell between <code><td></code> and <code></td></code> tags. This setting ignores other column settings such as Show Type Icon, href, format, and Alternate OID expression.</p> <p>icon--Used when the column values are shown as an icon. The setting "Column Icon" must be defined with the icon to be displayed. For more information on configuring a column using a program, see Column Values as Icons.</p> <p>checkbox--Used when the column values are check boxes shown grayed or not-grayed based on the access to the object in that row. This access can be based on business logic and defined in a JPO or it can be role-based and defined by assigning roles to the column. If the check boxes have no access restrictions, this setting is not required and you can create check boxes by passing in the parameter selection=multiple to emxTable.jsp. Also see Column Values as Check Boxes.</p> <p>separator--Used to define a column of white space between standard data columns. A separator is especially useful to separate two groups of columns.</p> <p>file--Shows which hyperlinks to a Quick File Access page listing files checked into or connected to the object. The optional Relationship Filter setting defines how to locate related files. If Common Components is not installed, this value is ignored.</p> <p>image--Used when the column values are images. When used, the column shows the small version of the primary image associated with the business object.</p>
Comparable	When the structure browser includes a structure compare feature, this setting allows you to include (true), or exclude (false) the column from the list of columns a user can select to Compare By.	false true (default)

Compare Report	When the structure browser includes a structure compare feature, this setting allows you to show or hide the column from the report results.	hide show (default)
Decimal Precision	Defines the decimal precision for all calculations for this column. The system properties setting <code>emxFramework.SBTTableCalculations.DecimalPrecision</code> defines the system-wide value; this setting overrides that value. You only need to use this setting if you want to use a value other than the system-wide value (defined in <code>emxSystem.properties</code>).	Any positive integer.
Diff Code	When multiple columns in a structure compare Complete Summary Report have different values, this setting defines what to show in the Diff Code column. The code value for the column with the lowest precedence shows in the Diff Code column. Multiple columns can have the same precedence, and if both columns have different values, both code values show in the Diff Code column.	<precedence:code value> 1:Attribute
Display Format	Specifies the date format for any column where the value of the format setting is "date." See Date/Time Fields in Forms and Tables . Not supported for non-object based structure browsers.	These are Java standard values of Date Format to display a date in a specific format. 3 - SHORT (12/12/52) 2 - MEDIUM (Dec 12, 1952) 1 - LONG (December 12, 1952) 0 - FULL (Tuesday, December 12 1952 AD) Set the default in <code>emxSystem.properties</code> : <code>emxFramework.DateTime.DisplayFormat=MEDIUM</code> . <code>emxSystem.properties</code> uses words, but the Display Format setting uses numbers.
Display Time	Controls whether the time is displayed along with the date for columns whose format is set to date. If no time zone preference is set, then the DateTime is shown in the browser's time zone. The time is shown in terms of <code>GMT+/- hh:mm</code> , (e.g., Saturday, August 21, 2004 12:45:00 PM <code>GMT-04:00</code>). To get the time in a format like EST or PDT, set the time zone preference to a specific zone. See Date/Time Fields in Forms and Tables . Not supported for non-object based structure browsers.	true false Default is set in <code>emxSystem.properties</code> for the property <code>emxFramework.DateTime.DisplayTime = false</code>
displayMode	Defines in which mode, Edit or View or Both, that this column is visible. When used, the visibility of the column is defined by the current mode and the value of this setting instead of an Access Expression. For example, if the structure browser is in view mode and this setting is Edit, then the column is not visible. This setting applies to columns used in the structure browser only; it does not work for columns in flat tables (<code>emxTable.jsp</code>).	Edit View Both (default)
Dynamic URL	When enabled, users can enter URLs or mxLink values and the values will display and function as hyperlinks.	enable (default) disable
Edit Access Function	Defines the method in the JPO specified by the Edit Access Program setting that determines access for each cell in the column. The method returns a <code>StringList</code> of boolean values for each cell (true/false) defining the edit access to that cell.	JPO method name
Edit Access Mask	Controls cell-level access in an editable table. The access check is done on all objects in one database call along with getting the column values. If any cell does not have the specified access, then the cell displays as read only.	Any access mask, for example: modify, connect
Edit Access Program	Defines the JPO to invoke that will determine whether the context user has edit access to individual cells in the	JPO name

	column.	
Editable	Use to indicate whether the column is displayed as editable or read only. Only applies for Edit mode. View mode ignores the setting.	true--Users can edit the column data when shown on the Edit mode. false (default)--Users cannot edit the column data when shown in the Edit mode. The column looks just like it does in View mode except it is never hyperlinked.
Export	Specifies whether column data is exported. Use this to change the export value on a column-by-column basis. By default, all column types except programHTMLOutput are exported. If you want to additionally include programHTMLOutput, or exclude other column types from the export, change this setting. See Column Values as Program Output .	true false
format	Specifies the display format. If at least one column has the format set to currency or UOM, the Conversion tool displays in the table page's page toolbar. When a user clicks the tool, the system opens a new window and displays all column data defined with format=currency and UOM to the currency and unit of measure selected in preferences. For information on the currency and unit of measure preferences, see About Unit of Measure Conversions and About Currency Conversions . Not supported for non-object based structure browsers.	date--Displays the column values as a date. Uses the tag lib "emxUtil:IzDate" to format the display. currency--Displays the column values as currency. UOM--Displays the column values as Unit of Measure and enables the Unit of Measure conversion interface. email--Displays the column values as an email address. When a user clicks the email address, the email editor configured in the client is presented. numeric--Displays the column values as numbers. To perform calculations or graphically analyze the data on a column of string attributes that have numerical values, numeric must be the column type user--Displays the user's name in the format "Lastname, Firstname".
function	The name of the method to call within the JPO program specified in the program setting. This method within the JPO is used to get the column values if the setting "Column Type" is set to "program" or "programHTMLOutput" or "checkbox". See Column Values as Program Output .	getAssignedBuyerDesk getPackageAccess getParentPart getCurrentState
Group Header	Defines header text to display over several consecutive columns. For example, if you want a group header over three consecutive columns, add this setting to each column and assign the same value for each. You may want to separate grouped columns using a separator column, which is just a column of white space. Add a separator using Column Type=Separator. To see an example, see Grouped Columns and Column Separator .	Static text or string resource id.
Group Name	Identifier used to collect columns into groups. All columns with the same Group Name will be grouped together. Used for merging cells as defined in Merged Cells .	Text String
Input Control Direction	Only used for structure browsers in Edit mode for columns that have an Input Type set to radiobutton or checkbox. Determines whether the field values display in a list (vertical) or on a single line (horizontal).	vertical (default) horizontal
Input Type	Only used for tables in Edit mode. Specifies the type of HTML control to display for user input. You can also designate the size of the input boxes to allow for appropriate spacing. This is important for short numeric entry fields. For fields that allow multiple selections, use checkbox instead of listbox (unless you implement a custom JPO). Multiple values are saved as a comma-separated list.	textbox--This is the default. The column has a single-line box for typing text. textarea--The column has a multi-line box for typing text. radiobutton--Shows a radio button next to each range value checkbox--Shows a checkbox next to each range value.

		<p>listbox--Provides a list of range values and users can select a single value. Users can select multiple values, but only one value is saved. If you implement a custom JPO, users can select multiple values and those values are saved as a comma-separated list.</p> <p>combobox--The column has a drop-down list of options and users can only select one value. Use combo boxes for attributes that have defined ranges and for attributes whose ranges are determined with a Range Helper URL. To see an example of a field with a combo box, see Column Values Editable from Combo Box, Values from JPO.</p>
Level	<p>Define the values to show in the Expand filter combo box.</p> <p>The values for this setting can be:</p> <ul style="list-style-type: none"> comma-separated list of positive integers All Specify <p>Text to be processed by the expandProgram JPO.</p> <p>You cannot use custom levels if the structure browser is filtered using relationship/type/direction.</p> <p>The All and Specify reserved keywords control specific levels of expansion.</p> <p>Use the Label setting to define the string resource or static text for the label that shows in front of the expand combo box; use the Registered Suite setting to define the application that defines the menu for the expand filter or the expand Program JPO.</p>	1,2,3,4,5 All Specify... Custom expand level (for example, End Item)
Mass Update	Used only for structure browsers in Edit mode. If set to false for the column, the column is not available to the Mass Update feature, but can be edited using the inline editing (one cell at a time). If set to true, the column is used by the Mass Update feature.	true false
Mouse Over Popup	<p>Enables or disables the mouse over popup DIV that displays the entire contents of the cell.</p> <p>When enabled, the popup shows in both View and Edit mode.</p>	enable disable (default)
OnChange Handler	The name of the JavaScript function called when the value in the cell is changed. You can provide a semicolon separated list of JavaScript functions.	<JavaScript function name>
OnFocus Handler	The name of the JavaScript function called when a cell is selected for edit. You can provide a semicolon separated list of JavaScript functions.	<JavaScript function name>
Popup Modal	If the setting Target Location is set to popup, the window can be configured as modal or non-modal.	true--the popup window is modal false--the popup window is non-modal If the setting is not specified, the window is non-modal.
Range Function	Use to specify the name of the method in the JPO specified in the Range Program setting. See the Range Program setting below for more information.	The name of a function in the Range Program JPO, such as: getAssignedRange getClassificationRange getPartUOM
Range Program	<p>Use to specify the name of a JPO that contains a method to get the column value ranges (choices). A range program is used only when the Input Type setting is combobox or popup.</p> <p>The corresponding method which runs to get the values of the column should not have any HTML code.</p>	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm

	To see an example of a field configured with a range program, see Column Values Editable from Combo Box, Values from JPO .	
*Registered Suite	<p>The application the column belongs to. The system looks for files related to the column in the registered directory for that application, which is specified in emxSystem.properties.</p> <p>Based on the application name, the system passes the following parameters in the href URL:</p> <ul style="list-style-type: none"> suiteKey emxSuiteDirectory StringResourceId 	<p>You must set the value without spaces, for example, EngineeringCentral or Framework. The value must be the suite name as defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. If the suite name starts with "eServiceSuite" then you can skip and just assign the remaining text. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite".</p> <p>In the href URL called when a user clicks the column data, the system passes a parameter called "suiteKey". The value for the parameter is the property name from emxSystem.properties that maps to the setting's value.</p>
Reload Function	<p>The name of the method in the JPO specified by the Reload Program setting that executes a cell reload.</p> <p>You also need to specify a value for the Reload Program setting.</p>	<JPO method name>
Reload Program	<p>The name of a JPO to invoke when this code in the structure browser is called:</p> <pre>emxReloadCell(<columnName>)</pre> <p>You also need to specify a value for the Reload Function setting.</p>	<JPO Name>
Root Label	<p>Determines the label for the root node(s). You can use static text, such as Search Results, or a string resource id. When using a string resource id, you can use any valid MQL expression, such as:</p> <pre>emxFramework.Common.RootLabel=\$<type>\$<name>:Root</pre> <p>If the expression does not apply to the root object (such as an attribute not associated with that object), the expression is ignored and the default label is used.</p> <p>If this setting is configured for a column not in the freeze pane, it is ignored.</p>	Search Results emxFramework.Common.RootLabel
Row Number	<p>If the column is defined with a Group Name setting, this setting defines whether this column displays in the first or second row of the group. Only 2 rows are permitted for a group.</p> <p>Used for merging cells as defined in Merged Cells.</p>	1 (default) 2
Row Span	<p>If this column is defined with a Group Name setting, defines how many rows each cell in this column will span. If set to 1, no spanning is done. If set to 2, the column value spans 2 rows. You cannot span more than 2 rows.</p> <p>Used for merging cells as defined in Merged Cells.</p>	1 2 (default)
Show Alternate Icon	<p>If the column value to display with the icon is different from the current row's object icon, set this to true. To get the right alternate icon, define the Alternate Type expression with the expression to obtain the object type.</p> <p>For more information, see Column Values Using Alternate OID in href and Select Expression.</p> <p>Not supported for non-object based structure browsers.</p>	true false
Show Type Icon	<p>If true, the column value displays with the object type icon as defined in the emxFramework.smallIcon property in emxSystem.properties. The icon displays to the left of the column data.</p> <p>If no property is defined for the type's icon, the system</p>	true false (default)

	<p>looks for a property defined for the parent type, then grandparent type and so on. If no property is defined for any type in the hierarchy, the system uses the default icon specified in the emxFramework.smallIcon.defaultType property.</p> <p>Not supported for non-object based structure browsers.</p>	
Sortable	Controls whether users can sort the table based on data in the table column. If a column is sortable, users can click the column heading to sort the table based on that column.	true (default) false
Sort Direction	Used only for fields of type combobox and listbox. The sort order for the option list.	ascending (default)--Sort a to z or 0 to n. descending--Sort z to a or n to 0. none--The option list is not sorted.
Sort Range Values	<p>Enables or disables the sorting of range values in combobox or listbox controls. When enabled, the list is sorted based on the datatype and using the direction defined by the Sort Direction setting. When disabled, no sorting is done on the list.</p> <p>In a drop-down, range values populated based on an attribute expression will be sorted based on the attribute's datatype. Range values populated using the Range Program and Range Function settings will be sorted alphanumerically.</p> <p>If the Range Program or Range Function returns numeric or date values, the alphanumeric sort will not be appropriate. The program or function should sort the values in the required order, and this setting should be disabled.</p> <p>This setting does not apply to fields or columns configured for attributes associated with a dimension. Dimension ranges are always sorted as alphanumeric in ascending order.</p> <p>This setting cannot be used in custom JSPs that use the editOptionList taglib. For this specific situation, you can use the sortType attribute for that tag.</p>	enable (default) disable
Sort Type	<p>Determines how the column is sorted. The MQL <code>sorttype</code> subclause must also be included.</p> <p><code>Sorttype</code> can be used on its own to sort numeric columns (<code>sorttype = numeric</code>)</p> <p>Or</p> <p><code>sorttype</code> can be used in conjunction with the Sort Type setting. Two examples:</p> <p>(<code>sorttype = other AND Sort Type = integer</code>)</p> <p>(<code>sorttype = other AND Sort Type = real</code>)</p>	date integer real AlphaNumericLarger--consider all numeric values as larger than the alphanumeric values AlphaNumericSmaller--consider all numeric values as smaller than the alphanumeric values
Style Column	Customizes the style for a column. The value must be a class defined in the dsecUITypeCustom.css. For the example style, the definition in the css file could be: <code>ColumnBackGroundColor { background-color : rgb(252,186, 186); }</code>	ColumnBackGroundColor
Style Function	The method in the Style Program JPO that defines the style per cell in the column.	The name of a function in the Style Program JPO, such as: getStyleInfo
Style Program	The JPO that contains methods to defines the style per cell in the column.	The name of a JPO, such as: AEFUtilStyleJPO
Target Location	Controls where the page specified in the href parameter appears or is targeted.	content--The page replaces the content frame. popup--Page appears in new window. Set the window modality with the Popup Modal setting. _top--Page replaces the entire body of the browser

		<p>window.</p> <p>listHidden--A hidden frame within the table frameset. Use this frame for Target Location for any processing in the context of a table.</p> <p>hiddenFrame--The frame called hiddenFrame is part of the top level Navigation window frameset and can be used to submit the Table frame and carry out background processing. This frame is not available for popup windows though so listHidden is a better frame to use. Set this value to any valid frame name that h is available to the table body frame.</p>
Type Icon Function	<p>Specifies the method in the JPO specified in the Type Icon Program setting that retrieves an icon to show in addition to the Alternate Icon or Type Icon (enabled by Show Alternate Icon or Show Type Icon settings).</p> <p>If both the Show Alternate Icon and the Show Type Icon settings are set to false, any value for this setting is ignored.</p>	Method Name
Type Icon Program	Defines the program that contains the function specified using the Type Icon Function setting.	JPO Name
Update Function	Specifies a method name in the JPO given in the Update Program setting.	<p>The name of a function in the Update Program JPO, such as:</p> <p>setAssignedBuyerDesk setPackage Access setPartClassification</p>
Update Program	<p>Specifies a JPO that contains a method to set the column value when the column is displayed on an Edit mode table and when the user clicks Done. This program is used only when the Field Type setting is program or programHTMLOutput.</p> <p>Using an update program is recommended only when the Field Type is not attribute or basic.</p> <p>See Using JPO to Update Table Values= for more information.</p>	<p>The name of a JPO, such as:</p> <p>SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm</p>
Validate	<p>Specifies the method to be invoked for validating any cell.</p> <p>This setting is deprecated. Existing structure browser columns that use this setting will still work, but new columns should use the OnChange setting instead.</p>	<p>The name of a method, such as:</p> <p>checkUniqueName</p>
Validate Type	Validates any field value with any specific characters defined in emxSystem.properties.	<p>Basic or Restricted--The field values are validated against the value of emxFramework.Javascript.BadChars.</p> <p>Name--The field values are validated against the value of emxFramework.Javascript.NameBadChars.</p>
Width	<p>Number of pixels to define the column width.</p> <p>When not defined the width is calculated based on the header text length so that the header text is not truncated.</p>	<p><column width in pixels></p> <p>150</p>
Window Height	<p>Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties.</p> <p>Use to define the height of the popup chooser window or window opened from any href link in a form field.</p>	Number of pixels, such as 700. The default is 600.
Window Width	<p>Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties.</p> <p>Use to define the width of the popup chooser window or window opened from any href link in a form field.</p>	Number of pixels, such as 700. The default is 600.

*Required Setting

Parameters for Structure Browser

This table describes the available parameters for table objects. Except for the table name, a table object is a set of column definitions so these parameters apply to columns within a table.

For specific instructions on how to create table objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Alt	<p>Used for structure browser columns when the Column Type is set to programHTMLOutput or image.</p> <p>The text defined for this parameter shows in the mouse-over popup DIV. If this parameter is not defined, the popup DIV shows the contents of the cell.</p>	Name of what the image or column data represents.

Subscriptions

Subscriptions allow users to subscribe to events that occur for an object, so that when the event occurs, a notification is sent to that user.

In this section:

- [About Subscriptions](#)
- [Subscription Options Dialogs](#)
- [Pushed Subscriptions](#)
- [Subscriptions for Administrative Types](#)
- [Subscriptions for Relationships](#)
- [Notifications for Subscriptions](#)
- [Email Listener](#)

About Subscriptions

Subscriptions use JPOs to notify subscribers when events occur.

See the *Live Collaboration Administrator's Guide* for instructions on customizing subscription events. Refer to the *Common Components User's Guide* for information on how a user interacts with subscriptions.

The subscription JSPs are stored in the components directory. You use these JSPs to control the subscription process:

- emxSubscriptionDialog.jsp: configures subscriptions on a single object
- emxSubscriptionPushDialog.jsp: allows one person to subscribe another person to events for a single object

Subscription events for an admin type follow the same inheritance as other admin types. For example, if a subtype does not have any subscription events defined, the system looks to its parent for events. If any type in the hierarchy has a subscription event defined, then that subtype supports subscriptions for that event.

Subscriptions can use the recursive option. When the subscription event is defined, it can be specified as recursive (Is Recursive=true) or not (Is Recursive=false). When false, the user does not have the option of subscribing using the recursive option. When true, the user has the option to subscribe with or without recursion.

When a user selects the recursion option when subscribing to an event, the user will be notified when that event takes place on the object or on the related objects (only those objects connected using the specified relationship).

For example, an administrator has defined a subscription event for the Folder Created event for the Workspace admin type. In the definition, these settings were specified:

- Is Recursive=true
- Recurse Relationship="from[\$<relationship_DataVaults>], from[\$<relationship_SubVaults>]"

When a user subscribes to the event AND chooses the "Apply to sub levels" option (enables the recursion function), then whenever a folder is created and connected to the Workspace, or in any folder connect to that folder.

The subscription dialog can be configured to support a single object or to support multiple objects, or to allow one person to push a subscription to another person.

The ENOVIA products install subscribable events as required for business processes. You can create additional subscribable events. See the *Live Collaboration Administrator's Guide* for instructions on using the Business Modeler to create a new subscribable event.

Subscription Options Dialogs

The dialog box users see when they Subscribe to an object is defined by emxSubscriptionDialog.jsp (in the components directory).

To define the notification sent when a subscribed event occurs, refer to the *Live Collaboration Administrator's Guide*.

emxSubscriptionDialog.jsp allows users to subscribe to a single object or multiple objects by the menu command passing the APPSubscription or APPSubscriptionMultiObject command as described in the following sections. When subscribing to multiple objects, they must be the same administrative type of object (that is, all Part objects, all ECR objects, but not a mix of objects).

The following topics are discussed:

- [Subscribing to a Single Object](#)
- [Subscribing to Multiple Objects](#)

Subscribing to a Single Object

When a single object ID is passed to emxSubscriptionDialog.jsp, the user can:

- Subscribe to any supported events
- Unsubscribe from any previously-selected events

When the objectID is passed to the JSP, it determines the state of subscribable events and shows events that have previously been selected as checked, and those that are not selected as unchecked. The user can toggle between checked/unchecked, and the state when the Done button is clicked determines whether or not the JSP will subscribe or unsubscribe the user from events.

For a single object, the command APPSubscription is passed and supports these parameters and settings:

Parameter	Description	Value
Name	Name of the command	APPSubscription
Label	Property key used to obtain the label from the properties file	emxComponents.Discussion.Subscribe
href	URL of the Subscribe command	#{COMPONENTS_DIR}/emxSubscriptionDialog.jsp
Setting	Description	Value
Target Location	Subscription dialog should always be in a popup window	popup
Submit	Specifies whether the system should send the object ID(s) (and relationship ID, if applicable) for the current page to the JSP specified in the href parameter. For subscriptions, this setting must be True. If not specified, the system assumes it is false.	true--Submits the IDs to the URL specified in the href parameter.
Access Expression	Controls access to the subscribable event. For details, see Controlling User Access to User Interface Components.	read
Access Function		
Access Mask		
Access Program		
Registered Suite	The application the column belongs to. The system looks for files related to the column in the registered directory for that application, which is specified in emxSystem.properties. For subscriptions, use Components.	Components
Event Type	The name of the event to use as value for the object's Event Type attribute.	Folder Content Changed Folder Created
History Bit	Same as the history events returned from the transaction trigger and is used to define the command name associated with a subscription event.	modify create
Access	Description	Value

Role	By default, this command is available to all users with the role Global user	Global User
------	--	-------------



Subscribing to Multiple Objects

A JPO can pass a comma-separated list of objectIDs to emxSubscriptionDialog.jsp. When multiple objectIDs are passed in, the page does not check the status (checked or unchecked) of events. No events will be shown as selected, even if some of them have previously been selected. The user cannot use this version of the dialog box to unsubscribe from events.

All Object IDs passed into the emxSubscriptionDialog.jsp page must be of the same admin type. Unlike the single subscription dialog box, this dialog box can only be used to subscribe to events; it cannot be used to unsubscribe from events.

Parameter	Description	Value
Name	Name of the command	APPSubscriptionMultiObject
Label	Property key used to obtain the label from the properties file	emxComponents.Discussion.Subscribe
href	URL of the Subscribe command	\${COMPONENTS_DIR}/emxSubscriptionMultiObjectsDialog.jsp

Setting	Description	Value
Target Location	Subscription dialog should always be in a popup window	popup
Submit	Specifies whether the system should send the object ID(s) (and relationship ID, if applicable) for the current page to the JSP specified in the href parameter. For subscriptions, this setting must be True. If not specified, the system assumes it is false.	true--Submits the IDs to the URL specified in the href parameter.
Row Select	Used only for links on table pages. Specifies whether the JSP specified for the href expects one, at least one, or no rows in the table (Middle Frame) to be selected. For subscribing to multiple objects, this setting must be set to "multi" and the setting Submit must be set to "true". If not specified, it is assumed to be none. The JSP specified for the href expects at least one row in the table (middle frame) to be checked. If no item is selected, then the user will see the following message: "Please select an Item". This message corresponds to the following key value in the emxFrameworkStringResource.properties file: emxFramework.Common.PleaseSelectitem	multi--Appropriate only for links that appear on table pages that have check boxes or radio buttons for each row.
Access Mask	Controls access to the subscribable event. For details, see Controlling User Access to User Interface Components.	read
Registered Suite	The application the column belongs to. The system looks for files related to the column in the registered directory for that application, which is specified in emxSystem.properties. For subscriptions, use Components.	Components
Event Type	The name of the event to use as value for the object's Event Type attribute.	string Folder Created
History Bit	Same as the history events returned from the transaction trigger and is used to define the command name associated with a subscription event.	modify create
Access	Description	Value
Role	By default, this command is available to all users with the role Global user	Global User

Pushed Subscriptions

The Subscriptions component allows a user to push a subscription to another user.

Refer to the *Schema Reference Guide* for the data model for pushed subscriptions.

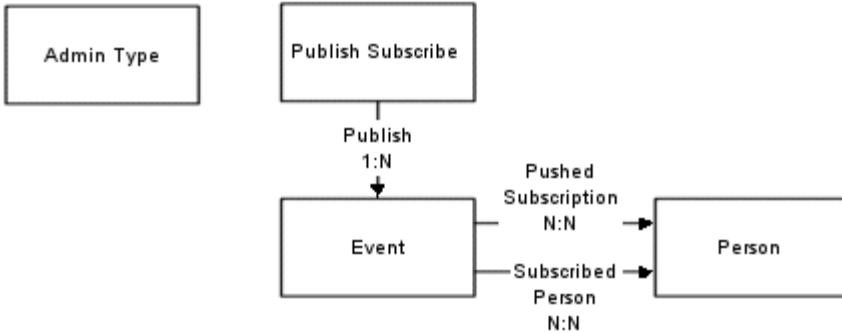
The push subscription functionality automatically picks up any subscribable events you define in addition to those provided out-of-the-box; you do not need any additional programming or configuration to implement pushed subscriptions. For instructions on using the UI to push a subscription, refer to the "Subscriptions" chapter of the *Common Components User's Guide*.

Subscriptions for Administrative Types

The API includes a set of methods that allow you to write custom functionality to allow end users to subscribe to an administrative type and not only to business objects. For example, a user could subscribe to the create event for a part, and then be notified whenever any part is created. The standard usage requires the user to subscribe to a specific object for specific events.

To define a type-level subscription event, see the instructions in the *Live Collaboration Administrator's Guide*. You must use the `Global=true` setting on the event command for the event to be configured on the type and not only on business objects.

For a type-level subscription, you must create a Publish Subscribe object using the symbolic name of the administrative type. The data model looks like this:



Although the Admin Type is not connected to the Publish Subscribe type, the relation is enforced because the name of the Publish Subscribe object must be the symbolic name of the administrative type.

For example:

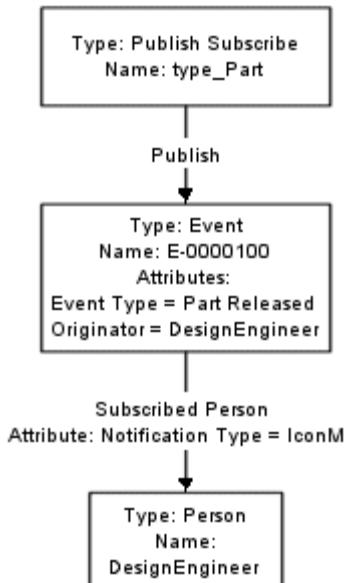
```
Type: Publish Subscribe  
Name: type_part
```

This Publish Subscribe business object would be connected to an Event object using the Publish relationship. The Event object must have the Event Type attribute set, for example Create, and the Originator (the person who subscribed to the event).

For notifications triggered by subscriptions, you can set the Notification Type attribute on the Subscribed Person relationship with one of these range values:

- Email
- IconMail
- Both (default)

This example shows the data model for a subscription:



In this example, whenever a part is released, the DesignEngineer receives a notification by IconMail.

Event subscriptions on administrative types follow the type hierarchy--that is, all subtypes of Part inherit the defined

subscription. A user can subscribe to a subtype when the parent type has an event command configured using the `Global=true` setting. By default, the Global setting is set to false.

Triggers also follow the type hierarchy. For example, if a Delete trigger on a the Part type is configured to send out a notification, then child objects of Part, such as Hardware Part inherit that trigger. Any triggers defined on the child object override an inherited trigger. When you define your triggers, make sure to include any logic from a parent type in the trigger for a child type, unless you explicitly do not want to retain that functionality.

See [Triggers](#), for more details.

To support this custom functionality, use the emxSubscriptionUtilBase JPO, which includes these methods:

Method	Description
subscribeOnObject	Creates subscription Event objects for the context user on a business object.
subscribeOnType	Creates subscription Event objects for the context user on an administrative type.
deleteSubscribeOnObject	Removes the subscription Event from a business object for the context user.
deleteSubscribeOnType	Removes the subscription Event from an administrative object for the context user.
getObjectNotifications	Queries all notifications (IconMails) generated by the object passed to the method. The method can filter the result based on the Event and Date range in which the notification was generated (Event and Date range must also be passed to the method). The results are returned as a maplist of IconMail details contained in a map of key/value pairs.
getTypeNotifications	Queries all notifications (IconMails) generated by the administrative type passed to the method. The method can filter the result based on the Event and Date range in which the notification was generated (Event and Date range must also be passed to the method). The results are returned as a maplist of IconMail details contained in a map of key/value pairs.

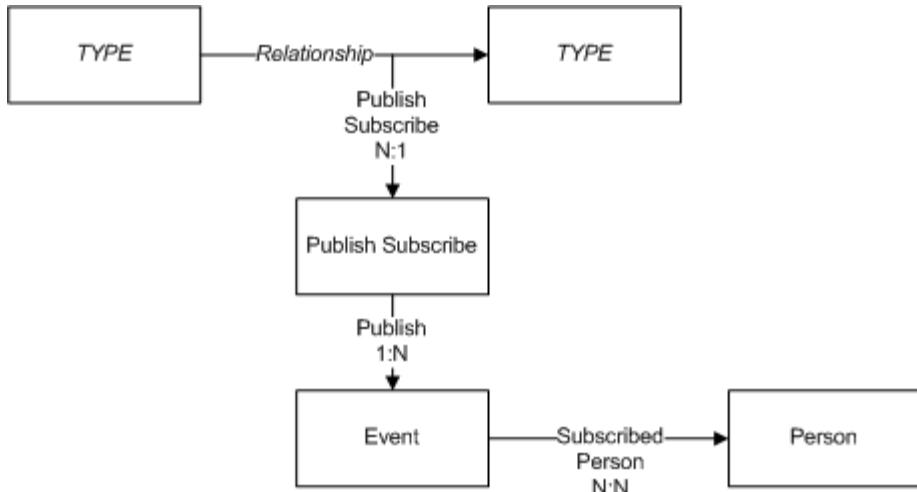
For details of the method signature and returned data, see the API documentation.

Subscriptions for Relationships

The API includes a set of methods that allow you to write custom functionality to allow end users to subscribe to a relationship event.

To define a relationship-level subscription event, see the instructions in the *Live Collaboration Administrator's Guide*. You must use the `Global=true` setting on the event command for the event to be configured on all relationships of a type and not just a single relationship connecting 2 business objects.

For a relationship-level subscription, you must create a Publish Subscribe relationship from another relationship to a Publish Subscribe object. The data model looks like this:



This Publish Subscribe business object is connected to an Event object using the Publish relationship. The Event object must have the Event Type attribute set, for example Create, and the Originator (the person who subscribed to the event).

To support this custom functionality, use the `emxSubscriptionUtilBase` JPO, which includes these methods. For details of the method signature and returned data, see the API documentation.

Notifications for Subscriptions

ENOVIA products include several JSPs to support subscriptions configured with notifications.

In this section:

- [About Notifications for Subscriptions](#)
- [Sample XML Template for Body Text](#)
- [Sample XML Template for Body HTML](#)
- [Sample JPO Code for Notification Attributes](#)

About Notifications for Subscriptions

ENOVIA products include several JSPs to support subscriptions configured with notifications.

Refer to the *Live Collaboration Administrator's Guide* for instructions on customizing notifications. When defining the attributes for a Notification object to be used with a subscription event, use these JSPs for the specified attributes:

Notification Object Attribute	Method in emxSubscriptionUtil	Argument Passed	Description
Dynamic To List	getSubscribersList		Retrieves all users who subscribed to the event.
Body Text	prepareBodyPlainText	Context context, String xmlDataString	Converts an XML string into plain text using an out-of-the-box XSL file
Body HTML	prepareBodyHTML	Context context, String xmlDataString	Converts an XML string into HTML format using an out-of-the-box XSL file

See the *Common Components User's Guide* for examples of the resulting Plain Text or HTML.

The Body Text and Body HTML attributes are optional for Notification objects, however, they are required when the notification is triggered by a subscription event. See the following sections for XML templates you can use to define the strings to pass to the listed methods.

When calculating the Dynamic To List, the method evaluates this expression:

```
from[$<relationship_PublishSubscribe>].to.from[$<relationship_Publish>][|to.attribute[$<attribute_EventType>]=='Folder Created'].to.from[$<relationship_SubscribedPerson>].to.name
```

where `Folder Created` is replaced with the appropriate event type.

Sample XML Template for Body Text

The `prepareBodyPlainText` method in `emxSubscriptionUtil` expects an `xmlString` as input. You can use this code as a template to construct your own XML input.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xmlstylesheet href="emxCreatPage.xsl" type="text/xsl"?>
<aef:mxRoot xmlns:aef="http://www.matrixone.com/aef">
    <aef:headerData>
        <aef:header>Some Header</aef:header>
        <aef:creatorText>Some Header</aef:creatorText>
    </aef:headerData>
    <aef:bodyData>
        <aef:sections>
            <aef:section>
                <aef:sectionHeader>section header</aef:sectionHeader>
                <aef:fields>
                    <aef:field>
                        <aef:label>field name</aef:label>
                        <aef:value>field value</aef:value>
                    </aef:field>
                </aef:fields>
            </aef:section>
            <aef:section>
                <aef:sectionHeader>section header</aef:sectionHeader>
                <aef:fields>
                    <aef:field>
                        <aef:label>field name</aef:label>
                        <aef:value>field value</aef:value>
                    </aef:field>
                </aef:fields>
            </aef:section>
            <aef:section>
                <aef:sectionHeader>section header</aef:sectionHeader>
                <aef:fields>
                    <aef:field>
                        <aef:label>field name</aef:label>
                        <aef:value>field value</aef:value>
                        <aef:oldValue>field's old Value </aef:oldValue>
                    </aef:field>
                </aef:fields>
            </aef:section>
        </aef:sections>
    </aef:bodyData>
    <aef:footerData>
        <aef:dataLines>
            <aef:dataLine>Text to display in the footer</aef:dataLine>
            <aef:dataLine>Text to display in the footer</aef:dataLine>
            <aef:dataLine>Text to display in the footer</aef:dataLine>
        </aef:dataLines>
        <aef:signature>Signature text</aef:signature>
    </aef:footerData>
</aef:mxRoot>
```

Sample XML Template for Body HTML

The `prepareBodyHTML` method in `emxSubscriptionUtil` expects an `xmlString` as input. You can use this code as a template to construct your own XML input.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xmlstylesheet href="emxCreatePage.xsl" type="text/xsl"?>
<aef:mxRoot xmlns:aef="http://www.matrixone.com/aef">
    <aef:headerData>
        <aef:header>SD 39404 moved to verification: IPGear: Shadow (SD 39401): Improve
error handling in restore server</aef:header>
        <aef:creatorText>SD was moved to verification by <a href="../common/
emxTree.jsp?objectId=54911.58640.61052.39527">Michael Brusser</a> </aef:creatorText>
    </aef:headerData>
    <aef:bodyData>
        <aef:sections>
            <aef:section>
                <aef:sectionHeader>Object Attachments</aef:sectionHeader>
                <aef:fields>
                    <aef:field>
                        <aef:label>Project</aef:label>
                        <aef:value>IP Gear</aef:value>
                    </aef:field>
                </aef:fields>
            </aef:section>
            <aef:section>
                <aef:sectionHeader>Changed Field(s)</aef:sectionHeader>
                <aef:fields>
                    <aef:field>
                        <aef:label>Fixed Release</aef:label>
                        <aef:value>2.1p5</aef:value>
                        <aef:oldValue>*BLANK*</aef:oldValue>
                    </aef:field>
                    <aef:field>
                        <aef:label>State</aef:label>
                        <aef:value>verification</aef:value>
                        <aef:oldValue>open</aef:oldValue>
                    </aef:field>
                    <aef:field>
                        <aef:label>Supporting Data</aef:label>
                        <aef:value>test_notes.txt text/plain</aef:value>
                        <aef:oldValue>*BLANK*</aef:oldValue>
                    </aef:field>
                </aef:fields>
            </aef:section>
            <aef:section>
                <aef:sectionHeader>Appended Field(s)</aef:sectionHeader>
                <aef:fields>
                    <aef:field>
                        <aef:label>Note Text</aef:label>
                        <aef:value>Added test-notes, moving to verification. Checking
in: restoreserver - New version: 1.12.1.4.2.3.1.33.1.1
                        Appended by Michael Brusser (michael) on 2006-03-23
10:18:58-0500:
                        </aef:value>
                    </aef:field>
                </aef:fields>
            </aef:section>
        </aef:sections>
    </aef:bodyData>
    <aef:footerData>
        <aef:dataLines>
            <aef:dataLine>Reply to this email to append directly to the note.</
aef:dataLine>
        </aef:dataLines>
    </aef:footerData>
</aef:mxRoot>
```

```
<aef:dataLine>Click here to <a href="../common/  
emxTree.jsp?objectId=54911.58640.61052.39127">View</a> the note.  
</aef:dataLine>  
</aef:dataLines>  
<aef:signature>  
    Email Notification from Synchronicity event trigger. (SyncDefect/39404)  
</aef:signature>  
</aef:footerData>  
</aef:mxRoot>
```

If the event is recursible but the subscription is not recursive, then the same expression is evaluated.

If the event is recursible and the subscription is recursive then the system gets all the parent objects and evaluates the above expression on all the parent objects to generate the consolidated Dynamic To List.

Sample JPO Code for Notification Attributes

When defining notification objects for subscription events, you can provide a JPO:method for many of the attributes.

This java program provides a sample JPO. Use the MQL insert program command to store code in the database.

```
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;
import matrix.db.Context;
import matrix.db.JPO;
import matrix.util.StringList;
// NotificationSample_mxJPO.java
//
// Created on May 23, 2006
//
// Copyright (c) 2005 MatrixOne Inc.
// All Rights Reserved
// This program contains proprietary and trade secret information of
// MatrixOne, Inc. Copyright notice is precautionary only and does
// not evidence any actual or intended publication of such program.
//
/**
 * @author bucknam
 *
 * The <code>NotificationSample_mxJPO</code> class contains sample methods
 * which can be used as attributes of a Notification object.
 *
 * @version AEF 10.7.0.0 - Copyright (c) 2005, MatrixOne, Inc.
 */
public class NotificationSample_mxJPO {
/**
 * Default JPO constructor
 * @param context the eMatrix <code>Context</code> object
 * @param args a String array containing input arguments
 *
 * @throws Exception if the operation fails
 * @since AEF 10.7.0.0
 */
public NotificationSample_mxJPO(Context context, String[] args)
throws Exception {
}
/**
 * This method is executed if a specific method is not specified.
 *
 * @param context the eMatrix <code>Context</code> object
 * @param args holds no arguments
 * @return an int 0, status code.
 * @throws Exception if the operation fails
 * @since AEF 10.7.0.0
 */
public int mxMain(Context context, String[] args) throws Exception {
    if (true) {
        throw new Exception(
            "must specify method on NotificationSample invocation");
    }
    return 0;
}
/**
 * This method represents a filter.
 *
 * @param context the eMatrix <code>Context</code> object
 * @param args contains a Map with the following entries:
```

```

*
*      id - a String holding the id of an object or relationship to be used for
*            evaluating selects embedded in the subject and message strings
*      idType - a String set to either "object" or "relationship"
*      notificationName - a String holding the name of the Notification object
*      payload - a Map containing information to pass on to helper JPO methods
* @return an int with 0 meaning let through
* @throws Exception if the operation fails
* @since AEF 10.7.0.0
*/
public static int evaluateFilter(Context context, String[] args)
    throws Exception {
    if (args == null || args.length < 1) {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    String id = (String) map.get("id");
    String idType = (String) map.get("idType");
    String notificationName = (String) map.get("notificationName");
    Map payload = (Map) map.get("payload");

    //int results = 1; //to block
    int results = 0; // to pass

    return results;
}
/***
* This method generates a "from" user.
*
* @param context the eMatrix <code>Context</code> object
* @param args contains a Map with the following entries:
*      id - a String holding the id of an object or relationship to be used for
*            evaluating selects embedded in the subject and message strings
*      idType - a String set to either "object" or "relationship"
*      notificationName - a String holding the name of the Notification object
*      payload - a Map containing information to pass on to helper JPO methods
* @return a String of the user agent
* @throws Exception if the operation fails
* @since AEF 10.7.0.0
*/
public static String generateFromAgent(Context context, String[] args)
    throws Exception {
    if (args == null || args.length < 1) {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    String id = (String) map.get("id");
    String idType = (String) map.get("idType");
    String notificationName = (String) map.get("notificationName");
    Map payload = (Map) map.get("payload");

    // String results = "User Agent";
    // Better to use symbolic notation...
    String results = "$<person_UserAgent>";

    return results;
}
/***
* This method generates a StringList of recipients.
*
* @param context the eMatrix <code>Context</code> object
* @param args contains a Map with the following entries:
*      id - a String holding the id of an object or relationship to be used for
*            evaluating selects embedded in the subject and message strings
*      idType - a String set to either "object" or "relationship"
*      notificationName - a String holding the name of the Notification object

```

```

*      payload - a Map containing information to pass on to helper JPO methods
* @return a Stringlist of the given list of items
* @throws Exception if the operation fails
* @since AEF 10.7.0.0
*/
public static StringList generateRecipients(Context context, String[] args)
    throws Exception {
    if (args == null || args.length < 1) {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    String id = (String) map.get("id");
    String idType = (String) map.get("idType");
    String notificationName = (String) map.get("notificationName");
    Map payload = (Map) map.get("payload");

    StringList results = new StringList(3);
    results.addElement("${<person_UserAgent>}");
    results.addElement("${<person_TestEverything>}");
    results.addElement("${<person_AdministrationUser>}");

    return results;
}
/***
* This method generates subject text.
*
* @param context the eMatrix <code>Context</code> object
* @param args contains a Map with the following entries:
*           id - a String holding the id of an object or relationship to be used for
*                 evaluating selects embedded in the subject and message strings
*           idType - a String set to either "object" or "relationship"
*           notificationName - a String holding the name of the Notification object
*           payload - a Map containing information to pass on to helper JPO methods
* @return a String of the subject
* @throws Exception if the operation fails
* @since AEF 10.7.0.0
*/
public static String generateSubjectText(Context context, String[] args)
    throws Exception {
    if (args == null || args.length < 1) {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    String id = (String) map.get("id");
    String idType = (String) map.get("idType");
    String notificationName = (String) map.get("notificationName");
    Map payload = (Map) map.get("payload");
    String fromAgent = (String) map.get("from");
    StringList replyTo = (StringList) map.get("replyTo");
    StringList objectIdList = (StringList) map.get("objectIdList");
    String basePropFile = (String) map.get("bundleName");
    String baseURL = (String) map.get("baseURL");
    String urlSuffix = (String) map.get("urlSuffix");
    Locale locale = (Locale) map.get("locale");
    String status = (String) map.get("status");
    StringList toList = (StringList) map.get("toList");
    StringList ccList = (StringList) map.get("ccList");
    StringList bccList = (StringList) map.get("bccList");

    String results = "This is a subject";

    return results;
}
/***

```

```

* This method generates body text.
*
* @param context the eMatrix <code>Context</code> object
* @param args contains a Map with the following entries:
*   id - a String holding the id of an object or relationship to be used for
*         evaluating selects embedded in the subject and message strings
*   idType - a String set to either "object" or "relationship"
*   notificationName - a String holding the name of the Notification object
*   payload - a Map containing information to pass on to helper JPO methods
* @return a String of the message body
* @throws Exception if the operation fails
* @since AEF 10.7.0.0
*/
public static String generateBodyText(Context context, String[] args)
    throws Exception {
    if (args == null || args.length < 1) {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    String id = (String) map.get("id");
    String idType = (String) map.get("idType");
    String notificationName = (String) map.get("notificationName");
    Map payload = (Map) map.get("payload");
    String fromAgent = (String) map.get("from");
    StringList replyTo = (StringList) map.get("replyTo");
    StringList objectIdList = (StringList) map.get("objectIdList");
    String basePropFile = (String) map.get("bundleName");
    String baseURL = (String) map.get("baseURL");
    String urlSuffix = (String) map.get("urlSuffix");
    Locale locale = (Locale) map.get("locale");
    String status = (String) map.get("status");
    StringList toList = (StringList) map.get("toList");
    StringList ccList = (StringList) map.get("ccList");
    StringList bccList = (StringList) map.get("bccList");

    String results = "This is a message \n Yes, it really is.';

    return results;
}
/***
* This method generates body HTML text.
*
* @param context the eMatrix <code>Context</code> object
* @param args contains a Map with the following entries:
*   id - a String holding the id of an object or relationship to be used for
*         evaluating selects embedded in the subject and message strings
*   idType - a String set to either "object" or "relationship"
*   notificationName - a String holding the name of the Notification object
*   payload - a Map containing information to pass on to helper JPO methods
* @return a String of the message body in HTML format
* @throws Exception if the operation fails
* @since AEF 10.7.0.0
*/
public static String generateBodyHTML(Context context, String[] args)
    throws Exception {
    if (args == null || args.length < 1) {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    String id = (String) map.get("id");
    String idType = (String) map.get("idType");
    String notificationName = (String) map.get("notificationName");
    Map payload = (Map) map.get("payload");
    String fromAgent = (String) map.get("from");
    StringList replyTo = (StringList) map.get("replyTo");

```

```

        StringList objectIdList = (StringList) map.get("objectIdList");
        String basePropFile = (String) map.get("bundleName");
        String baseURL = (String) map.get("baseURL");
        String urlSuffix = (String) map.get("urlSuffix");
        Locale locale = (Locale) map.get("locale");
        String status = (String) map.get("status");
        StringList toList = (StringList) map.get("toList");
        StringList ccList = (StringList) map.get("ccList");
        StringList bccList = (StringList) map.get("bccList");

        StringBuffer sb = new StringBuffer();
        sb.append("<HTML>\n");
        sb.append("<HEAD>\n");
        sb.append("<TITLE>\n");
        sb.append("Title Text\n");
        sb.append("</TITLE>\n");
        sb.append("</HEAD>\n");
        sb.append("<BODY>\n");
        sb.append("<H1>" + "Header Text" + "</H1>" + "\n");
        sb.append("This is an HTML message");
        sb.append("</BODY>\n");
        sb.append("</HTML>\n");
        String results = sb.toString();

        return results;
    }
}

/**
 * This method modifies the given map.
 *
 * @param context the eMatrix <code>Context</code> object
 * @param args contains a Map with the following entries:
 *     id - a String holding the id of an object or relationship to be used for
 *           evaluating selects embedded in the subject and message strings
 *     idType - a String set to either "object" or "relationship"
 *     notificationName - a String holding the name of the Notification object
 *     payload - a Map containing information to pass on to helper JPO methods
 * @return an int with 0 meaning continue with sending mail
 * @throws Exception if the operation fails
 * @since AEF 10.7.0.0
 */
public static Map preprocess(Context context, String[] args)
    throws Exception {
    if (args == null || args.length < 1) {
        throw (new IllegalArgumentException());
    }
    Map map = (Map) JPO.unpackArgs(args);
    String id = (String) map.get("id");
    String idType = (String) map.get("idType");
    String notificationName = (String) map.get("notificationName");
    Map payload = (Map) map.get("payload");
    String fromAgent = (String) map.get("from");
    StringList replyTo = (StringList) map.get("replyTo");
    StringList objectIdList = (StringList) map.get("objectIdList");
    String basePropFile = (String) map.get("bundleName");
    String baseURL = (String) map.get("baseURL");
    String urlSuffix = (String) map.get("urlSuffix");
    Locale locale = (Locale) map.get("locale");
    String status = (String) map.get("status");
    StringList toList = (StringList) map.get("toList");
    StringList ccList = (StringList) map.get("ccList");
    StringList bccList = (StringList) map.get("bccList");
    String subject = (String) map.get("subject");
    String messageText = (String) map.get("messageText");
    String messageHTML = (String) map.get("messageHTML");
}

```

```
Map results = new HashMap();
// Need only pass back those items used to formulate the email message
results.put("from", fromAgent);
results.put("replyTo", replyTo);
results.put("toList", toList);
results.put("ccList", ccList);
results.put("bccList", bccList);
results.put("subject", subject + " modified");
results.put("messageText", messageText + " modified");
results.put("messageHTML", messageHTML);

//results.put("status", "Do Not Send"); //to block
results.put("status", ""); // to pass

return results;
}

}
```

Email Listener

The email listener works with subscriptions and discussions. If the Business Administrator sets the emxComponents.CaptureEmailAutoCCSubscriptionEnabled to true, then the system creates a discussion when any recipient of a notification replies to the notification.

In this section:

- [About the Email Listener](#)
- [The mailListener.xml File](#)
- [Preprocessing Email Messages](#)
- [Invoking the Email Listener JPO](#)
- [Troubleshooting the Email Listener](#)

About the Email Listener

The email listener works with subscriptions and discussions. If the Business Administrator sets the `emxComponents.CaptureEmailAutoCCSubscriptionEnabled` to true, then the system creates a discussion when any recipient of a notification replies to the notification.

The system also subscribes all of the people included on the notification to the discussion, and forwards the reply to those people. Each time anyone uses email to reply to a message in the discussion, that email message is forwarded to all other subscribed people. In this way, users can participate in a discussion without logging into the ENOVIA system.

The email listener is a stand-alone Java program that polls the inbox of one or more specified users and retrieves emails from the mail server using pop3 or imap protocols. When an incoming message is found, the email listener parses the message and forwards relevant information to a processing JPO (part of the Business Process Services) as xml. The email listener uses a Java Mail API for communications with the mail server.

The email listener daemon can process multiple inboxes using a separate thread for each inbox. It supports Text and HTML email formats but does not process attachments. Any attachments to emails are dropped, and that action is logged (if a log file name is defined in the `mailListener.xml` file).

The email listener includes these functions

- Use of a `mailListener.xml` file to determine which inboxes to monitor. You should configure the `mailListener.xml` file before you can use Email Listener.
- Preprocessing of the email message
- Invoking a JPO and passing the xml-processed email message

The mailListener.xml File

A mailListener.xml file defines the information required for each defined inbox.

Data Item	Description	Accepted Values/Examples
Protocol	The protocol used to communicate with the email server for user-to-mailbox access.	pop3 imap
UserName	The user name defined on the email server	MailBoxUser
Password	The user's password to access email	MailBoxPassword
eMail Host	Name of the email host to be monitored	
Polling Interval	Length of time between polling of the inbox, in milliseconds	
Processing JPO name	Name of the JPO that will process parsed email messages	
JPO Method name	Method in the JPO that will process parsed email messages	mxMain
MatrixUrl	URL to the ENOVIA Live Collaboration Server The MatrixUrl should point to a running server (RMI or WebServer). RMI in process mode is not supported by MailListener.	

Preprocessing Email Messages

When the email listener daemon finds a message in a monitored inbox, it preprocesses the message into xml format and then passes the xml to the JPO as the first element of the arguments array. The message object itself cannot be passed to JPO.invoke because it does not implement Serializable.

The parsed xml format looks like this:

```
<MxMessage>
  <version>1</version>
  <To>To addresses</To>
  <From>From address</From>
  <ReplyTo>ReplyTo address</ReplyTo>
  <Cc>Cc addresses</Cc>
  <Bcc>Bcc addresses</Bcc>
  <Subject>testing</Subject>
  <Text>Message text</Text>
  <HtmlText>HtmlTest</HtmlText>
  <Sent>JavaDate in Milliseconds</Sent>
  <Received>JavaDate in Milliseconds</Received>
</MxMessage>
```

Invoking the Email Listener JPO

The email listener connects to the ENOVIA Live Collaboration Server using an anonymous context.

The processing JPO:

- Is executed using a JPO invoke call
- Has a super user assigned to it
- Can reset context to any user without a password

The JPO method has this signature:

```
public int mxMain(Context context, String []args) throws  
MatrixException
```

The first element of the arguments array is the XML preprocessed message as described above.

If the invocation of the JPO:

- Is successful (no thrown exception), the message is deleted from the inbox.
- Throws an exception, the message remains in the inbox and will be processed again.

To support this JPO, the list person MQL command includes an email selectable. This command returns a list of persons with email addresses that match the provided pattern:

```
list person email EMAIL_ADDRESS_PATTERN
```

The processing JPO uses the result to identify a person in the database based on the email address in the from field.

Troubleshooting the Email Listener

You can troubleshoot email listener related problems by enabling the SMTP debug. To enable SMPT debug, add the `ematrix.smtp.debug` as java startup parameter.

The following sample shows a `mailListener.sh` file with instructions on how to enable SMPT debug.

```
Dematrix.smtp.debug = true;
#####
# Set up Java parameters
# For smtp debugging add -Dematrix.smtp.debug=true to
JAVA_OPTIONS
#
#####
JAVA_OPTIONS="-Xss512k -Xms256m -Xmx256m"
```

The email listener uses `java.util.logging` package for logging. The `java.util.logging` package is part of JDK starting from version 1.4.

The `mailListener.xml` also defines the file name pattern for runtime logging using the XML tag `matrixMailListener log file`. The file name pattern accepts flags defined in `FileHandler` documentation found in <http://java.sun.com/j2se/1.4.2/docs/api/java/util/logging/FileHandler.html>.

If you do not define the file name, logging does not take place.

Sample SMTP debug

```
DEBUG: setDebug: JavaMail version 1.3.3
polling ...
DEBUG: getProvider() returning
javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.
POP3Store,Sun Microsystems, Inc]
DEBUG POP3: connecting to host "engmaill.matrixone.net", port
110, isSSL false
S: +OK Qpopper (version 4.0.5) at engmaill starting.
C: USER matrixeng
S: +OK Password required for matrixeng.
C: PASS engmatrix
S: +OK matrixeng has 1 visible message (0 hidden) in 7974
octets.
C: STAT
S: +OK 1 7974
C: NOOP
S: +OK no big woop
C: TOP 1 0
S: +OK Message follows
Received: from mail.matrixone.com (matrixmail.matrixone.net
[10.1.16.122])
      by engmaill.matrixone.net (8.12.10+Sun/8.12.9) with
ESMTP id k63FpJch021
372
      for <matrixeng@engmaill.matrixone.net>; Mon, 3 Jul 2006
11:51:19 -0400 (
EDT)
```

Automatic Type Ahead

For Form and Table components, automatic type ahead can be delivered automatically for Type, Person, and Organization fields without the need for writing a specific-purpose JPO.

In this section:

- [About Automatic Type Ahead](#)
- [Defining Automatic Type Ahead for a Person Field/Column](#)
- [Defining Automatic Type Ahead for an Organization](#)
- [Defining Automatic Type Ahead Using Advanced Search](#)

About Automatic Type Ahead

This feature provides type-ahead functionality for by setting the Range Href setting for the field to emxFullSearch.jsp. In addition, a pre-defined type-ahead feature is provided for Type, Person, and Organization fields. The functionality is similar to the Type Ahead options that require a custom JPO, but without requiring developers to write a custom JPO.

The following topics are discussed:

- [Types of Type Ahead Configurations](#)
- [Advanced Search Used for Type Ahead](#)
- [Pre-Defined Type Ahead Choosers](#)

Types of Type Ahead Configurations

Type ahead can be achieved using any of these methods:

- Custom JPO that retrieve the list of possible values for a field. See [Entering Field Values Using Type Ahead](#).
- Advanced Search that uses emxFullSearch.jsp to retrieve possible values for a field. See [Advanced Search Used for Type Ahead](#).
- Pre-defined Choosers for Person, Organization, or Type fields. See [Pre-Defined Type Ahead Choosers](#).

If a field is defined with both a custom JPO and advanced search for Type Ahead, then the custom JPO is used.



Advanced Search Used for Type Ahead

Developers can use the advanced search function, emxFullSearch.jsp, to quickly configure a form field or table column for type ahead.

This method of implementing relies on the indexed searching provided by the Advanced Search feature. Only those fields defined in config.xml for indexing can be used for automatic type ahead. When the user starts typing in the field, they get

the same initial suggestion list that they would get if they clicked the button and opened the advanced search page. This feature does not work with choosers that do not use Advanced Search.

To implement type ahead using advanced search, you need to provide values as follows:

- **Type Ahead Mapping** setting on the field (default = `NAME`). Can be any field defined in config.xml. If you want the type ahead vaules to map to multiple fields, then include a comma-separated list of fields as the value for the Type Ahead Mapping setting, such as:
`LASTNAME,FIRSTNAME,USERNAME`
- **Type Ahead Validate** setting on the field (default = `false`). When the user enters a value, they can type something other than one of the values returned by the type ahead suggestion list. This could result in invalid data being entered. If you change this setting's value to `true`, then the user must select a value from the suggestion list.
- **emxFramework.FullTextSearch.TypeAhead.Suggestion.Limit** property in emxSystem.properties (default = 50). The value for this property limits how many objects will be shown in the suggestion list. If the suggestion list exceeds this value, **More...** is shown in the table. The user can continue to type characters to narrow down the suggestion list.

If a field has a value defined for **Type Ahead Mapping** and the **TypeAhead Program** and **Type Ahead Function** settings, then the JPO method overrides the mapping to the config.xml file.

If you need to restrict the suggestion list based on the current context, such as restricting task owners to those users who have access to the project the task belongs to, you can pass macros. For example:

```
emxFullSearch.jsp?field=TYPE=Person:PROJECT_ID=${<setting_name>}&table...
```

where `setting_name` is the name of the setting on the form or column. This setting value is a select expression to retrieve the runtime value for the query based on the object Id. See [Selectables](#) and [Macros](#) for more details.

If the user could be expected to enter a value that may be in more than one field, you can provide an OR list of fields. For example, when selecting a Person, the user could start typing that Persons' first name, their last name, or perhaps their user name. To configure the field for this type of entry, you can use this format (OR is indicated by the double-pipe symbols):

```
emxFullSearch.jsp?  
field=TYPE=Person:LASTNAME=*<enteredtext>* | FIRSTNAME=*<enteredtext>* | USERNAME=*<enteredtext>*
```

where `<enteredtext>` is the text the user typed into the field.

These properties in emxSystem.properties file used by the Type Ahead using JPO feature are also used by the automatic Type Ahead feature:

- `emxFramework.TypeAhead`
- `emxFramework.TypeAhead.TunProgramCharacterCount`
- `emxFramework.TypeAhead.SavedValues.Limit`

If you change the values for these properties, both the advanced search and custom JPO type ahead features are affected.



Pre-Defined Type Ahead Choosers

The most commonly searched-for values when filling in dialogs are Person, Organization, and Type. Business Process Services allows a developer to configure type ahead by setting the value for the Range Href setting for the field to one of these values:

- PERSON_CHOOSER
- ORGANIZATION_CHOOSER
- emxTypeChooser.jsp

See [Configurable Type Chooser](#) for details on configuring the field to use this chooser. When the user starts typing in the field, they get the same initial suggestion list that they would get if they clicked the button and opened the type chooser dialog. The suggestion list is internationalized based on the user's browser language setting.

The PERSON_CHOOSER href maps to this:

```
 ${COMMON_DIR}/emxFullSearch.jsp?field=TYPE=type_Person:STATE=Active&table=AEPersonChooserDetails&selecton=single&submitURL=${COMMON_DIR}/emxPersonChooserSelectProcess.jsp
```

In addition, the [Type Ahead Mapping](#) setting on the field should be set to `FIRST_NAME, LAST_NAME, NAME`.

The ORGANIZATION_CHOOSER href maps to this:

```
 ${COMMON_DIR}/emxFullSearch.jsp?  
field=TYPE=type_Organization:STATE=Active&table=AEOrganizationChooserDetails&  
selection=single&submitURL=${COMMON_DIR}/emxOrganizationChooserSelectProcess.jsp
```

In addition, you must set the value for the [Type Ahead Mapping](#) setting on the field to `Name`.

If the application requires restricting the value for the field more than that defined by the above hrefs, you can also define a value for the [Additional Query](#) setting on the field or column using this syntax:

```
"Additional Query" = <FieldName1>=<select expression1>:<FieldName1>=<select expression2>:
```

The Field Names used in the expression must be the names of fields configured for indexing in config.xml. See [Selectables](#) for details on select expressions for the current object.

Defining Automatic Type Ahead for a Person Field/Column

Automatic type ahead can easily be defined for a Person form field or table column without requiring a custom JPO.

Before you begin: The config.xml file must be configured to index Person objects.

1. In Business Modeler, locate the needed web form or table and open it for editing.
2. Click the name of the form field or table column that contains a Person. The field/column could be named Owner, Assignee, and so on.
3. Click **Edit** next to the list of fields/columns.
4. Click the **Link** tab.
5. In the RangeHref text box, enter `PERSON_CHOOSER`.
6. Define a value for the Type Ahead Mapping setting:
 - a. Click the **Settings** tab.
 - b. In the **Name** text box, enter `Type Ahead Mapping`.
 - c. In the **Value** text box, enter the Name of a field or a series of fields defined in config.xml, such as `FIRST_NAME, LAST_NAME, NAME`.
 - d. Click **Set**.
7. If you need to restrict the list of Person's based on the application context, follow these steps (still on the Settings tab):
 - a. In the **Name** text box, enter `Additional Query`.
 - b. In the **Value** text box, enter the fields and select expressions to use to restrict the suggestion list in this format:
`<FieldName1>=<select expression1>:<FieldName1>=<select expression2>`
where the field names are fields defined in config.xml, and the select expression is a valid expression for the field as defined in [Selectables](#).
 - c. Click **Set**.
8. Click **OK** to save the field/column changes.
9. Click **Edit** to save the wWeb form/table.
10. If necessary, update the Business Process Services properties that work with Type Ahead:
 - a. In a text editor, open emxSystem.properties for editing.
 - b. Edit the values for these properties as needed:
 - `emxFramework.FullTextSearch.TypeAhead.Suggestion.Limit`
 - `emxFramework.TypeAhead`
 - `emxFramework.TypeAhead.RunProgram.CharacterCount`
 - `emxFramework.TypeAhead.SavedValues.Limit`

See the *Live Collaboration Administrator's Guide* for details on editing emxSystem.properties.

Defining Automatic Type Ahead for an Organization

Automatic type ahead can easily be defined for an Organization form field or table column without requiring a custom JPO.

Before you begin: The config.xml file must be configured to index Organizations.

1. In Business Modeler, locate the needed web form or table and open it for editing.
2. Click the name of the form field or table column that contains an Organization. The field/column could be named Organization, Business Unit, Supplier, and so on.
3. Click **Edit** next to the list of fields/columns.
4. Click the **Link** tab.
5. In the RangeHref text box, enter `ORGANIZATION_CHOOSER`.
6. Define a value for the Type Ahead Mapping setting:
 - a. Click the `Settings` tab.
 - b. In the **Name** text box, enter `Type Ahead Mapping`.
 - c. In the **Value** text box, enter `NAME`.
 - d. Click **Set**.
7. If you need to restrict the list of Person's based on the application context, follow these steps (still on the Settings tab):
 - a. In the **Name** text box, enter `Additional Query`.
 - b. In the **Value** text box, enter the fields and select expressions to use to restrict the suggestion list in this format:
`<FieldName1>=<select expression1>:<FieldName1>=<select expression2>`
where the field names are fields defined in config.xml, and the select expression is a valid expression for the field as defined in [Selectable](#)s.
 - c. Click **Set**.
8. Click **OK** to save the field/column changes.
9. Click **Edit** to save the web form/table.
10. If necessary, update the Business Process Services properties that work with Type Ahead:
 - a. In a text editor, open emxSystem.properties for editing.
 - b. Edit the values for these properties as needed:
 - `emxFramework.FullTextSearch.TypeAhead.Suggestion.Limit`
 - `emxFramework.TypeAhead`
 - `emxFramework.TypeAhead.RunProgram.CharacterCount`
 - `emxFramework.TypeAhead.SavedValues.Limit`

See the *Live Collaboration Administrator's Guide* for details on editing emxSystem.properties.

Defining Automatic Type Ahead Using Advanced Search

Automatic type ahead can easily be defined for a form field or table column without requiring a custom JPO. You can use `emxFullSearch.jsp` and the properties described in this task to set up automatic type ahead.

Before you begin: The config.xml file must be configured to index the object/field being configured for type ahead.

1. In Business Modeler, locate the needed web form or table and open it for editing.

2. Click the name of the needed field or column.

3. Click **Edit** next to the list of fields/columns.

4. Click the **Link** tab.

5. In the RangeHref text box, enter `emxFullSearch.jsp`.

See [Full-Text Search](#) for more details of URL parameters that can be included in the RangeHref.

6. Define a value for the Type Ahead Mapping setting:

a. Click the **Settings** tab.

b. In the **Name** text box, enter `Type Ahead Mapping`.

c. In the Value text box, enter `NAME`.

d. Click **Set**.

7. Define a value for the Type Ahead Validation setting (if you do not want to use the default value of false)

a. In the **Name** text box, enter `Type Ahead Validation`.

b. In the Value text box, enter `true` or `false` depending on your business requirements.

c. Click **Set**.

8. Click **OK** to save the field/column changes.

9. Click **Edit** to save the web form/table.

10. If necessary, update the Business Process Services properties that work with Type Ahead:

a. In a text editor, open `emxSystem.properties` for editing.

b. Edit the values for these properties as needed:

- `emxFramework.FullTextSearch.TypeAhead.Suggestion.Limit`
- `emxFramework.TypeAhead`
- `emxFramework.TypeAhead.RunProgram.CharacterCount`
- `emxFramework.TypeAhead.SavedValues.Limit`

See the *Live Collaboration Administrator's Guide* for details on editing `emxSystem.properties`.

DesignSync File Access

ENOVIA products are integrated with DesignSync to improve the way files, folders, and modules are version controlled from ENOVIA Live Collaboration. This section shows how to customize applications to work with DesignSync.

In this section:

- [DesignSync File Access \(DSFA\) Overview](#)
- [DSFA Operations](#)
- [FCS and DesignSync](#)
- [DSFA Selectables](#)
- [History for DesignSync File Access](#)
- [Find the ENOVIA Object ID from DesignSync](#)
- [Synchronize ENOVIA Updates with DesignSync](#)
- [Troubleshooting DSFA](#)

DesignSync File Access (DSFA) Overview

ENOVIA products are integrated with DesignSync to improve the way files, folders, and modules are version controlled from ENOVIA Live Collaboration. DesignSync is a version control system managing the changes to a collection of files, folders, and modules over time.

To access DesignSync from an ENOVIA application, System Administrators must define a DesignSync store in ENOVIA Live Collaboration. See the *System Manager Guide* for instructions on creating a DesignSync store.

DesignSync File Access from ENOVIA Live Collaboration is supported via the Semiconductor Accelerator only.

A DesignSync store represents a DesignSync server and is used to associate DesignSync files, folders, and modules (a versioned set of managed files, folders and collection objects in DesignSync) with ENOVIA Live Collaboration business objects. Business objects are connected to files, folders, and modules in a DesignSync server using an ENOVIA product. Business objects that use DesignSync stores use vcfile, vcfolder, and vcmodule (vc = version control) operations to map DesignSync files, folders, and modules to them.

You can create, checkin, checkout, view, and remove files and folders from the DesignSync server using ENOVIA products. File and folder operations do not access the DesignSync server directly, but operate on the connections that associate (connect) the files and folders in DesignSync to business objects in ENOVIA. DesignSync tracks the changes to file and folder versions; ENOVIA Live Collaboration records the metadata related to file and folder operations in business objects for reference.

DesignSync stores use the HTTP or HTTPS protocol to connect to DesignSync. The FCS Studio Customization Toolkit classes may be used to checkin new files to DesignSync, but Matrix Navigator or Web Navigator with out-of-the-box FCS cannot be used for checkin.

In order to use HTTPS, you must configure SSL on the DesignSync Data Manager, as described in the DesignSync INSTALL.txt file, and install the digital certificate onto the Web server used for the ENOVIA Live Collaboration Server. You install the certificate by copying it to the application server machine and using the JVM tools to install the certificate into the JVM trusted keystore. Some servers, for example, WebLogic, require additional configuration on the web server in order to recognize the digital certificate. For more information on installing digital certificates, consult the documentation for your web server.

DSFA Operations

You can connect, disconnect, copy, modify, and delete files, folders, and modules.

In this section:

- [About DSFA Operations DSFA](#)
- [Connect to a File in DesignSync](#)
- [Connect to a Folder in DesignSync](#)
- [Connect to a Module in DesignSync](#)
- [Copy Files to Other DesignSync Stores](#)
- [Modify a Connection](#)
- [Disconnect from a File, Folder, or Module in DesignSync](#)
- [Delete a File or Module from a DesignSync Store](#)

About DSFA Operations DSFA

A business object is connected with a file, folder, or module in a DesignSync server using the connect businessobject command. This connection allows for DesignSync file access from ENOVIA applications and custom programs.

You can also create an empty connection, where a business object is associated with a non-existent file, folder, or module. Later, when you check in a file to the connection, the file is associated with the appropriate business object. For every new connection created, an index number unique to that connection is generated. The information needed to identify a version of a file, folder, or module in a configuration is stored in ENOVIA Live Collaboration while the actual file, folder, or module and all its metadata is stored in the DesignSync server.

In ENOVIA Live Collaboration, files, folders, and modules in a DesignSync server are referred to as vcfiles, vcfolders, and vcmodesules respectively.

When a business object in a DesignSync store is cloned, the connections associated with the object are copied over to the newly-created business object. When a business object is revised, the connections are copied only if the object has the option to copy any files checked into it. The following sections explain how to connect and disconnect business objects to files, folders, or modules in DesignSync.

When connecting to DesignSync files from ENOVIA applications, you can specify a checkintag string for the DesignSync files. For DesignSync folder connections, you can specify checkintag for the files contained in the DesignSync folder.

The checkin tag is automatically generated during DesignSync checkin and when using the copyfromstore functionality. It is used by the system to find relevant versions of checked in or copied files and are useful when you need to check out file versions that are not the latest versions (that is file versions that are not at the end of a branch). The ability to specify the checkin tag during vcfile or vcfolder connect operation is useful when migrating data (files and folders) from ProjectSync to ENOVIA Live Collaboration.

The checkintag string you specify is stored in ENOVIA Live Collaboration as metadata.

Connect to a File in DesignSync

If you have checkin access to a business object, you can use the Connect Businessobject command to connect (associate) a new or an existing version of a file in DesignSync (vcfile).

The following topics are discussed:

- [Connect Command](#)
- [Specify Version](#)
- [Config Clause](#)
- [Store Clause](#)
- [Format Clause](#)
- [Lock Clause](#)
- [Description Clause](#)
- [Complete or Incomplete Clause](#)
- [Checkintag Clause](#)

Connect Command

This MQL command shows the syntax for connecting to a DSFA object.

```
connect businessobject BO_NAME vcfile path PATH VERSION [VCFILE_ITEM {VCFILE_ITEM}];
```

- `BO_NAME` is the business object you are associating with the vcfile.
- `PATH` is the string that identifies the path of folders down to the last level filename of the vcfile relative to the path defined in the store identifying the server.
- `VERSION` is a means to identify the version of the vcfile.
- `VCFILE_ITEM` is a connect businessobject clause providing information about the connection you are creating. The different clause options are

```
config NAME  
store NAME  
format NAME  
lock  
description STRING  
[in]complete  
checkintag STRING
```



Specify Version

The combination of all the information (must be 255 characters or less) provided for a version is called the version data of the connection. The different ways to specify the version of a vcfile are as follows:

```
selector [string]  
versionid [string]  
versiontag [string]  
branchid [string][qualifier string]  
branchtag [string][qualifier string]
```

Selector, versionid, versiontag, branchid and branchtag are all specifier types. The string you provide for each of the specifier types is the specifier. The specifier types branchid and branchtag have optional qualifiers. Each specifier type and the optional qualifiers are described below.

- `selector` is the string that identifies a version in the format used by the DesignSync applications. For example, the `1.4 gold` (as a version tag), and `Rel2:Latest` are possible selectors.
- `versionid` is a string identifying a particular version of the file in DesignSync. For example, `1.3` and `1.3.1.5` are valid version ids of a file in DesignSync.
- `versiontag` is a string identifying a tag applied to a particular version of a file. For example, `gold` might be a valid example of a version tag for a file.
- `branchid` is the string identifying the branch of the vcfile using a DesignSync branch id. For example the strings `1` and `1.2.1` are potential branch ids of a file.

The main branch `Trunk` is designated with `branchid 1` and files on this branch have version numbers `1.1, 1.2, 1.3`, and so on.

- `branchtag` is a string identifying a tag applied to the branch of the file in DesignSync. For example, `branchtag Release-Final` is a valid example of a branch tag for a file.

A branchtag should always end with a colon to differentiate a branchtag from a versiontag. If you do not provide the colon, you get an error.

`branchid` and `branchtag` have optional qualifiers to identify the version of the branch picked out by the connection. The qualifier can have the following possible values: `Latest` (used as a qualifier only, cannot be used by itself) or a valid DesignSync time format. If no qualifier is provided, the default value of `Latest` is appended as the qualifier (for example `Trunk:Latest`).

If you provide a selector, you do not need to provide the qualifier.



Config Clause

This clause is the name of a folder configuration being associated with the file connection. This value can be used with the `select vcfolder` allowing you to navigate to the folder containing the file.



Store Clause

This clause is the name of the DesignSync store (representing a DesignSync server). If the name of a store is not provided, it is obtained from the business object policy.



Format Clause

This clause is the name associated with the vcfie.



Lock Clause

A lock can be used only for files that physically exist in a DesignSync store. It cannot be used for a non-existent file.

When you lock a file in a DesignSync server, the business object associated with the file is not locked. ENOVIA Live Collaboration keeps no record of DesignSync files locked by users. That information is kept in DesignSync and can be retrieved when needed.



Description Clause

The description is a string holding information about the connection you are creating.



Complete or Incomplete Clause

This clause is a "completeness" flag with the possible values of complete and incomplete. The value of complete, the default value, indicates that the path clause contains the last-level of the filename. If the flag is set to incomplete during checkin, the last-level of the filename is appended to the path name and the flag is reset to complete.



Checkintag Clause

This clause is a string holding the autotag (the version tag) value.

All the strings issued with this command except path and description should contain 255 characters or less.

Connect to a Folder in DesignSync

If you have checkin access to a business object, you can use the Connect Businessobject command to connect (associate) a new or an existing configuration of a folder in DesignSync (vcfolder).

This MQL shows how to connect to a folder:

```
connect businessobject BO_NAME vcfolder path PATH config NAME [VCFOLDER_ITEM {VCFOLDER_ITEM}];
```

- **BO_NAME** is the business object you are associating with the vcfolder.
- **PATH** is the string that identifies the path of folders leading to the vcfolder relative to the path defined in the store identifying the server.
- **config NAME** is the name of the configuration in the DesignSync server which contains the folder.
- **VCFOLDER_ITEM** is a connect businessobject clause providing information about the connection you are creating. The different clause options are:

```
store NAME
```

```
format NAME
```

```
description STRING
```

Each clause is described in the previous section.

The following error message may occur after attempting to connect a business object in ENOVIA Live Collaboration with a file or folder in a DesignSync server:

```
Error: #1900068: add store failed Error: #1900346:  
DesignSync server at host faure.matrixone.net port 30080 could not be  
found.
```

This error indicates the lack of an authenticated connection to the DesignSync server. This may be the result of a network problem, an incorrect server url, a server not running, or the ENOVIA Live Collaboration user defined is not authenticated by that server.

Connect to a Module in DesignSync

If you have checkin access to a business object, you can use the Connect Businessobject command to connect (associate) a new or an existing configuration of a module in DesignSync (vcmodule):

This MQL command connects to a module:

```
connect businessobject BO_NAME vcmodule path PATH VERSION [VCMODULE_ITEM];
```

- **BO_NAME** is the business object you are associating with the vcmodule.
- **PATH** is the string that identifies the path of folders leading to the vcmodule relative to the path defined in the store identifying the server.
- **VERSION** is a means to identify the version of the vcmodule. See [Specify Version](#) above.
- **VCMODULE_ITEM** is a connect businessobject clause providing information about the connection you are creating. The different clause options are:

```
store NAME  
description STRING  
[in]complete
```

Each clause is described in [Connect to a File in DesignSync](#).

Copy Files to Other DesignSync Stores

The **Copy Businessobject** command is used to move the file(s) of a vcmodule connection or vcfolder connection to another DesignSync repository pointed to by a DesignSync store in ENOVIA Live Collaboration.

This MQL command copies to a DSFA store:

```
copy businessobject BO_NAME vcmodule to store NAME path STRING [VERSION];
```

Or

```
copy businessobject BO_NAME vcfolder to store NAME path STRING config NAME;
```

- **BO_NAME** is the name of the business object that is connected to the vcmodule or vcfolder from which you are copying files.
- **store NAME** is the DesignSync store to which you are copying over the files.
- **path STRING** is the string that identifies the path of folders leading to vcmodule or vcfolder from which you are copying the files. If this path does not exist in the destination store, it is created during the copy process.
- **config NAME** is the name of the configuration in the DesignSync server which contains the folder.
- **VERSION** is the version of the vcfile you are copying. The VERSION determines what happens if the file already exists in the destination DesignSync store. If the file already exists, the VERSION is used to determine what branch to append the file as a new version. A new version is only created if the file differs from the version that is currently the last one on the branch. If any file being copied already exists at the target location, the same rule applies with the module version determining the branch.

Modify a Connection

After a vcfolder, vcmodule connection is defined, you can change the definition with the `Modify Businessobject` command. This command lets you add or remove defining clauses and change the value of clause arguments. If you have checkin access, you can modify existing vcconnections.

This MQL command modifies a connection:

```
modify businessobject BO_NAME vcconnection [index NUMBER] [VC_MOD_ITEM];
```

Or

```
modify businessobject BO_NAME vcmodule path PATH [VC_MOD_ITEM];
```

- `BO_NAME` is the name of the business object associated with the connection .
- `index NUMBER` is an identifier number unique among all connections. Each connection is assigned one when created and this value never changes. Unless the index number of a connection is specified, the modify command modifies ALL connections.
- `PATH` is the string that identifies the path of folders leading to the vcmodule relative to the path defined in the store identifying the server.
- `VC_MOD_ITEM` indicates the modifications you can make. The modifications you can make correspond to the different clauses you can define when you create the connection. The modifications that can be made for all three connection types are specified in the following table. Some of the modifications described in the table are not possible for all three connection types.

Modify Businessobject Clause	Specifies that...
<code>store NAME</code>	The name of the DesignSync store (representing a DesignSync server) is changed to the NAME given.
<code>format NAME</code>	The name associated with the connection is changed to the NAME given.
<code>path STRING</code>	The string that identifies the path of folders leading to vcfolder, or vcmodule relative to the path defined in the store identifying the server is changed to the STRING given.
<code>description STRING</code>	The description is changed to the STRING given.
<code>sparse true</code>	Attributes for business objects of this type that are not populated when the object is create/modified are not instantiated (not part of the business object's Oracle tables).
<code>sparse false</code>	All attributes defined for the type are instantiated for all business objects of the type, even if the attributes are not populated.
<code>versionid STRING</code>	The string identifying a particular version of a vcfolder or vcmodule is changed to the STRING given.
<code>versiontag STRING</code>	The string identifying the tag applied to a particular version of a vcfolder or vcmodule is changed to the STRING given.
<code>add remove versiontag STRING</code>	The version tag applied to a particular version of a vcfolder or vcmodule is either added to or removed from the vcfolder or vcmodule. If the add operation is not successful, an error is returned. Errors are not returned for the remove operation regardless of outcome.
<code>branchid STRING</code>	The string identifying the branch of the vcfolder or vcmodule is changed to the STRING given.
<code>branchtag STRING</code>	The string identifying the tag applied to the branch of the vcfolder or vcmodule is changed to the STRING given.
<code>add remove branchtag STRING</code>	The branch tag applied to a particular version of a vcfolder or vcmodule is either added to or removed from the vcfolder or vcmodule. When you remove a branch tag from a version, you remove the tag from the branch of that version. If the add operation is not successful, an error is returned. Errors are not returned for the remove operation regardless of outcome.
<code>selector STRING</code>	The string identifying the branch and version of the vcfolder or vcmodule is changed to the STRING given.

<code>qualifier STRING</code>	The string identifying the version of the branch picked out by the connection is changed to the STRING given.
<code>config STRING</code>	The name of the folder configuration being associated with the file connection is changed to the STRING given. This value can be used with the select vcfolder command allowing you to navigate to the folder containing the file.
<code>[in]complete</code>	The "completeness" flag is changed to complete or incomplete.
<code>checkintag STRING</code>	The string holding the autotag (the version tag value) is changed to the STRING Given.
<code>[un]lock</code>	The vcfile is either locked or unlocked.
<code>copyfromstore STORE_NAME path STRING VERSION </code> Or, use this command when copying vcfolders: <code>copyfromstore STORE_NAME path STRING config NAME ;</code>	<p>The vcfiles or vcfolders from one DesignSync store are copied to another store.</p> <p>The BO_NAME and the index NUMBER specified in the command are the business object connected to one or more vcfiles or vcfolders of the store you are copying to and the number of the connection to a vcfile or vcfolder in the store you are copying to. If the index number is not specified, the command applies to all vcconnections associated with the business object.</p> <p>The STORE_NAME is the DesignSync store from which you are copying over the vcfile. The path STRING is the string that identifies the path of folders leading to vcfile from which you are copying the vcfile. If this path does not exist in the destination store, it is created during the copy process. The VERSION is the version of the vcfile you are copying. The different ways to specify the version of a vcfile are described in Specify Version. The config NAME is the name of the configuration of the vcfolder you are copying.</p> <p>In the destination store, an auto-generated checkin tag is applied to the latest version of the file that is copied over. If the file already exists at the destination position determined by the specifier on the vcconnection, and its content is identical to that of the file being copied, then the new file is not copied. The existing file at the destination gets the checkin tag placed on the matching version.</p>

All the strings issued with this command except path and description should contain 255 characters or less.

Disconnect from a File, Folder, or Module in DesignSync

If you have checkin access, you can disconnect a business object from a vcfile, vcfolder, or vcmodule using the Disconnect Businessobject command.

This MQL command disconnects:

```
disconnect businessobject BO_NAME vcconnection [index NUMBER] [unlock];
```

Or

```
disconnect businessobject BO_NAME vcmodule path PATH;
```

- **BO_NAME** is the name of the business object that is connected to the vcfile, vcfolder, or vcmodule.
- **index NUMBER** is an identifier number unique among all connections. Each connection is assigned one when created and this value never changes.
- **PATH** is the string that identifies the path of folders leading to the vcmodule relative to the path defined in the store identifying the server.

Note: The disconnect command operates on connections specified by the index number. If an index number is not provided, the command disconnects all file connections.

- **unlock** is an optional keyword used when disconnecting a vcfile that unlocks a file locked in DesignSync. You can disconnect a connection to a vcfile that is in a locked state in the DesignSync server. The vcfile remains locked after the disconnection.

Delete a File or Module from a DesignSync Store

If you have checkin access, you can remove a specific version of a vcf file or vcmodule from a DesignSync store using the Delete Businessobject command. You can also remove the entire vcf file or vcmodule.

This MQL command deletes a file or module:

```
delete businessobject BO_NAME vcconnection [index NUMBER] [version];
```

Or

```
delete businessobject BO_NAME vcmodule path PATH [version];
```

- **BO_NAME** is the name of the business object that is connected to the vcf file or vcmodule.
- **index NUMBER** is an identifier number unique among all connections. Each connection is assigned one when created and this value never changes. The index number identifies the file or module to be deleted.
- **PATH** is the string that identifies the path of folders leading to the vcmodule relative to the path defined in the store identifying the server.
- **version** indicates that the system should remove the version pointed to by the connection and leave all the other versions.

If version is not specified, the entire file is deleted.

Under certain circumstances, the version cannot be deleted. For example, you cannot delete the first version of a branch. In such cases, an error is returned. You cannot remove a module version on the DesignSync server. When deleting a vcmodule, the command only deletes the business object and NOT the module from DesignSync.

FCS and DesignSync

When interacting with a DesignSync server, the FCS uses http and https protocols. The matrix.db.BusinessObject class in the Studio Customization Toolkit uses the FCS to checkin and checkout files and folders from a DesignSync server.

In this section:

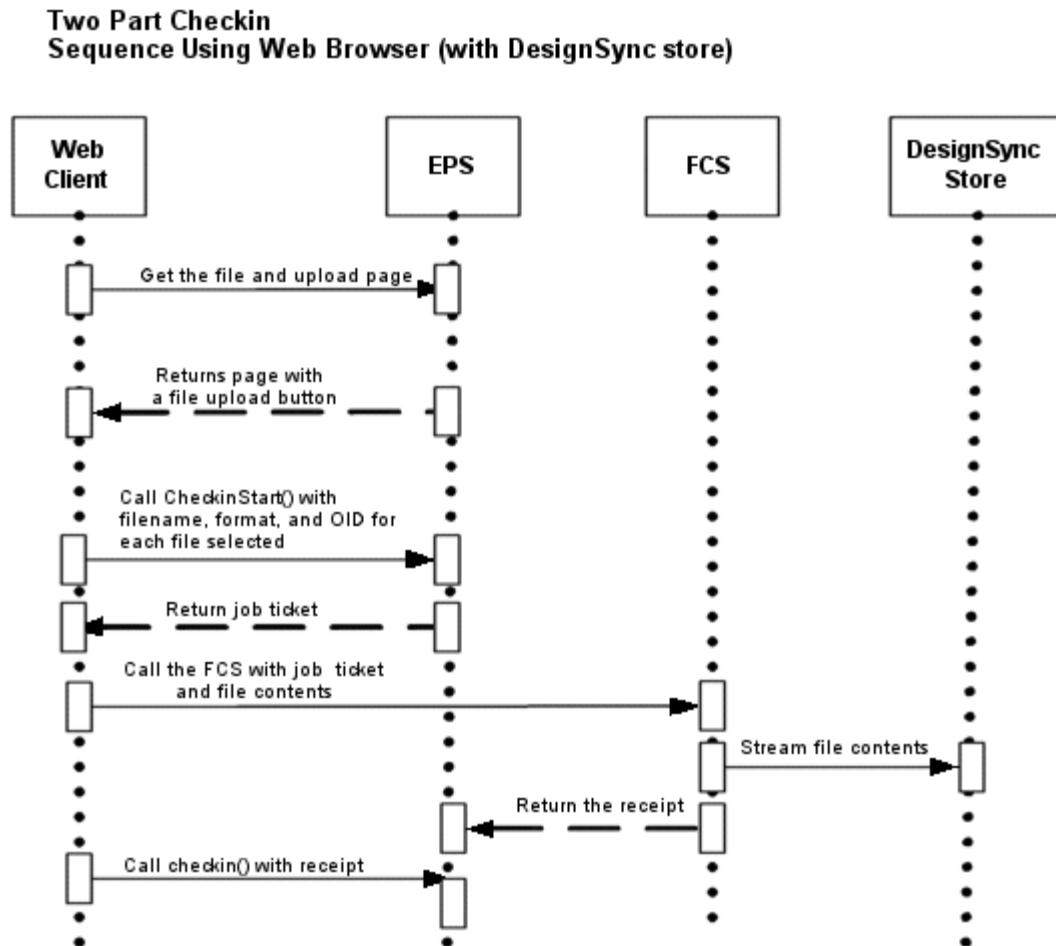
- [Check in DesignSync Files](#)
- [Check in DesignSync Folders](#)
- [Check in Physical Data to DesignSync Modules](#)
- [Check Out DesignSync Files](#)
- [Check Out DesignSync Folders](#)
- [Check out a DesignSync Module](#)
- [Write Two-Part Checkin of Vcfile and Vcfolder](#)
- [Write Checkout of Vcfile and Vcfolder](#)

Check in DesignSync Files

For DesignSync files and folders, FCS supports the two-part checkin. This includes getting a ticket from the MCS, then executing a checkinStart and then executing checkin and checkinEnd.

For information on two-part checkin, see [Two-Part Checkins](#).

The following diagram illustrates two-part checkin of a DesignSync file:



When checking in files to a DesignSync store, check and action triggers are supported. Macros available to these triggers include VCINDEX, which contains the index number of the vcconnection (the connection associating the business object to the DesignSync file). The index number provides all other information about the connection including the format and filename. Therefore, the checkin event macros FORMAT and FILENAME are not available in this case.

The following information is required to check in a vcfile:

- The VERSION of the file. Checking in a new file or a newer version of a file succeeds only if the version of the file is specified as: branch:Latest or by no qualifier, (which is interpreted to mean Latest) or a selector which resolves to a branch and the latest version on it.

Note: Check in fails and returns an error if you checkin a new physical file to an existing file in the DesignSync server where the version specification does not resolve to branch:Latest.

- An optional comment describing your check in. If provided, it gets associated in DesignSync with the version of the file you are checking in.

All vcfiles that get checked in get an auto-generated CHECK_IN tag. This tag is returned to the FCS and to the MCS via the receipt.

When you check out a vcfile, it is locked by the DesignSync server. If other users try to check out the file, they are displayed a warning. The business object connected to the file remains unlocked.

For information on how to write a client that performs two part checkin of a vcfile, see [Write Two-Part Checkin of Vcfile and Vcfolder](#).

Check in DesignSync Folders

You can check in folders and files to existing vcfolders. The configuration of the vcfolder folder connection determines the branch where its files get checked in. When checking in a new vcfolder, the folder (and its files) are created in the DesignSync store. During subsequent check in to the folder, only the files in the folder that are modified and new files added to the folder are copied to the DesignSync server.

To check in a vcfolder, compress the contents of the folder into a zip or gzip tar file and check in the compressed file. Each individual file in this compressed file has a path that is relative to the path of the vcfolder.

If a file's content matches any of the versions on the branch to which it is being copied, it does not get copied to the DesignSync server. Even if a file does not get copied, all files in the compressed file are considered part of the vcfolder connection and will be included in any subsequent checkout of the vcfolder.

During check in, enter an optional comment describing your check in. If provided, it gets associated with the version of the file you are checking in.

All files in the vcfolder that get checked in get an auto-generated CHECK_IN tag. This tag is returned to the FCS and to the MCS via the receipt.

For information on how to write a client that performs two part checkin of a vcfolder, see [Write Two-Part Checkin of Vcfile and Vcfolder](#).

Check in Physical Data to DesignSync Modules

You can check in a zip or gzip tar file to a DesignSync module.

Each individual file in this compressed file has a path that is relative to the path of the module. The folders and files you checkin may or may not already exist in DesignSync. If a folder (and its files) does not already exist, it is created by DesignSync and the new files are checked in as new files for the DesignSync system.

When you check in a zip or gzip tar file, the operation creates a new module version in the DesignSync system. The contents of the new module version will match the contents of the zip or gzip tar file. The name of the new Module version is made available to you in the receipt.

During check in, enter an optional comment describing your check in. If provided, it gets associated with the module.

Check Out DesignSync Files

The checkout operation for DesignSync files follow the same sequence as checkout of ordinary files.

As with checkin, the check and action triggers are available. Macros available to these triggers include VCINDEX, containing the index number of the vcconnection (the connection associating the business object to the DesignSync file). The index number provides all other information about the connection including the format and filename. Therefore, the checkin event macros FORMAT, FILENAME_ORIGINAL and FILENAME are not available.

For information on how to write a client that performs a checkout of a vcfolder, see [Write Checkout of Vcfolder](#).

To check out a different version of a vcfolder than the one it specifies, you need to pass in a string that has one of these keywords:

- selector
- versionid
- versiontag
- branchid
- branchtag

followed by a colon, followed by the appropriate specification of a selector, versionid, versiontag, branchid, or branchtag. If branchid or branchtag is used, it can optionally be followed by a colon and a qualifier (where the qualifier is in the form allowed when specifying a vcfolder connection).

Check Out DesignSync Folders

A DesignSync folder is checked out as a compressed file with a .tgz extension.

For information on how to write a client that performs a checkout of a vcfolder, see [Write Checkout of Vcfile and Vcfolder](#).

When you check out a folder, only the folder and the files that were initially checked into the folder get checked out. This is because when you check in a new DesignSync folder, The ENOVIA product uses the CHECK_IN tag to identify the folder and the files that are checked in. During a check out, the ENOVIA product looks for the folder and the files that have the CHECK_IN tag and checks them out. All new files checked into the folder directly from DesignSync are not visible to the ENOVIA products as they do not have the CHECK_IN tag. These are left out by ENOVIA products.

To checkout all files inside a folder including those checked in using DesignSync, create a new connection from the business object connected to the vcfolder and use this new connection to check out the contents of the folder.

Do not use this connection to check into the folder, or the set of files that can be checked out will be restricted.

Check out a DesignSync Module

When you check out a vcmodule, it is returned to you as a zip or gzip tar file.

You can also choose to check out a different version than the one specified by the vcmodule. To check out a different version, you need to pass in a string that has one of these keywords:

- selector
- versionid
- versiontag
- branchid
- branchtag

followed by a colon, followed by the appropriate specification of a selector, versionid, versiontag, branchid, or branchtag. If branchid or branchtag is used, it can optionally be followed by a colon and a qualifier (where the qualifier is in the form allowed when specifying a vcfile connection).

You cannot lock a module during a checkout operation.

Write Two-Part Checkin of Vcfile and Vcfolder

To write a client that performs a two-part checkin of a vcfile or vcfolder, use the doIt() method of the HttpVcCheckinStart and HttpCheckinEnd classes at the appropriate times as indicated in the sequence diagram.

The class com.matrixone.fcs.http.HttpVcCheckinStart is derived from the parent class com.matrixone.fcs.mcs.CheckinStart. There is no derived class required for com.matrixone.fcs.http.HttpCheckinEnd; used it as done for FCS checkin for captured stores as described in [File Collaboration Server](#).

The HttpVcCheckinStart.doIt() method is shown below.

```
public static TicketWrapper doIt(  
    Context ctx,  
    java.lang.String processingPage,  
    java.lang.String targetPage,  
    java.lang.String errorPage,  
    java.lang.String[] boids,  
    long[] vcIndexList,  
    java.lang.String[] comments,  
    java.lang.String[] unlock,  
    javax.servlet.http.HttpServletRequest req,  
    javax.servlet.http.HttpServletResponse res)  
    throws MatrixException
```

The arguments are:

Argument	Description
Ctx	The ENOVIA context.
processingPage	The page to use for the checkin end. This is the page where the checkin.end() is called from. This page should perform any user processing. Users' parameters will be available on this page.
targetPage	The final destination page.
errorPage	The error page.
boids	An array of business object ids for the business objects whose associated files or folders are being checked in.
vcIndexList	An array of vcconnection indexes for the files and folders being checked in.
comments	An array of comments, one for each of the vcfiles or vcfolders being checked in.
unlock	An array of flags that indicates whether the files in the DesignSync server are to be unlocked on checkin.
req	An Http request which must have a matching params of boid, vcIndex, comment, and unlock. If the param counts do not match, an error is thrown.
res	Http Servlet response

Write Checkout of Vcfile and Vcfolder

To write a client that performs a checkout, use the doIt() method of the `HttpVcCheckout` class passing in various parameters.

The doIt() method is available in the `com.matrixone.fcs.http.HttpVcCheckout` package.

The `HttpVcCheckout` class provides two version of the doIt() method as shown below.

```
public static TicketWrapper doIt(
        Context ctx,
        java.lang.String[] boids,
        java.lang.String[] fileNames,
        java.lang.String[] formats,
        java.lang.String[] locks,
        long[] vcIndexList,
        java.lang.String[] versions,
        java.lang.String[] path,
        boolean zipOutput,
        java.lang.String outFileName,
        java.lang.String errorPage,
        javax.servlet.http.HttpServletRequest req,
        javax.servlet.http.HttpServletResponse res)
        throws MatrixException
```

The arguments are:

Argument	Description
Ctx	The ENOVIA product context.
boids	An array of business object ids for the business objects whose associated files or folders are being checked in.
fileNames	An array containing the names of the files to checkout.
formats	An array containing the formats of the files to checkout.
lock	An array containing the lock state of the files after the checkout is complete. The files are locked in the DesignSync store. The business objects associated with the files do not get locked during file check out.
vcIndexList	An array of vcconnection indexes for the various files and folders being checked in.
versions	An array of alternate version specification. For vcfile checkout, the format to specify the alternate version is: <code>specifier type@specifier@qualifier</code> (if needed). For information on specifier type, specifier and qualifier, see the explanation of VERSION in Connect to a File in DesignSync . For vcfolder checkout, specify the configuration name. If you provide an empty string, the file or the folder pointed to by the connection is checked out.
paths	An array containing the paths to the files. The path to prepend to a file in the zip file on checkout.
zipOutput	
OutFileName	This is what the file will be referred to locally after the checkout.
errorPage	The URL location to the error page to display if the checkout operation fails.
req	An Http request which must have a matching params of boid, vcIndex, comment, and unlock. If the param counts don't match, an error is thrown.
res	Http Servlet response

This function can be used to checkout more than one vc connection as well as files from captured stores, or to checkout a specific file of a vcfolder connection. This is why the arguments take arrays of information. Each arg, which is an array, must

have the same number of entries as every other one. Entries with the same index refer to the same captured file or vc connection to check out. When specifying a captured file, the vcindex should be zero. When specifying a vc connection (so that vcindex > 0), the format is irrelevant. For a vc connection, the filename is only relevant when checking out a specific file of a vcfolder connection.

DSFA Selectables

The selectables for DSFA can be used with queries on vcfolder and vcmodule connections. ENOVIA products access the DesignSync server to retrieve any needed information depending on the selectable used.

In this section:

- [Business Object Selectables](#)
- [Selectable Fields for Vcfolder and Vcmodule](#)
- [Selectable Fields for Abstract Vcfolder and Vcmodule](#)

Business Object Selectables

This table lists business object selectable fields.

Selectable Fields for Business Objects		
Field	Description	Output
vcfile	Provides information about the vcfile connection of the business object.	The output depends on the various forms in which this field is specified. These forms are described below
vcfolder	Provides information about the vcfolder connection of the business object.	The output depends on the various forms in which this field is specified.
vcmodule	Provides information about the vcmodule connection of the business object.	The output depends on the various forms in which this field is specified.

- For `vcfile`, `vcfolder`, or `vcmodule`, returns TRUE if any vcfile, vcfolder, or vcmodule connection respectively exists, else returns FALSE.
- For `vcfile[]`, `vcfolder[]`, or `vcmodule[]`, returns TRUE if the string inside the brackets is the index of a vcfile, vcfolder, or vcmodule connection respectively, else returns FALSE.
- For `vcfile[.*]`, `vcfolder[.*]` or `vcmodule[.*]`, the system checks that the string inside the brackets is the index of a vcfile, vcfolder, or vcmodule connection respectively. If true, applies the next keyword to that connection. The type of value returned depends on the keyword. For a list of keywords, see the selectable fields for that connection below.
- For `vcfile.*`, `vcfolder.*` or `vcmodule.*`, the next keyword is applied in turn to each vcfile, vcfolder, or vcmodule connection respectively. The type of value returned depends on the keyword. Each line of output includes the index of the vcfile, vcfolder, or vcmodule connection respectively to which it applies. The index is enclosed in brackets on the left-hand side.

Selectable Fields for Vcfile, Vcfolder and Vcmodule

Selectables for vcfiles, vcfolders and vcmodes can return ENOVIA metadata or data held in the DesignSync system. The following table lists selectables that return ENOVIA metadata.

ENOVIA metadata Selectable Fields

Field	Description	Output
abstract.* (vcfile or vcmodule only)		Works only with a "..". Applies the next select keyword to the abstract vcfile or vcmodule of the current vcfile or vcmodule version picked out by this connection
author (vcfile or vcmodule only)	Name of the person who checked in the vcfile or vcmodule version.	Returns the name of the owner (author) of the file or module.
branchid (vcfile or vcmodule only)	Branch id of a vcfile or vcmodule version.	Returns the id of the branch of the current file or module picked out by the connection.
branchtag[] (vcfile or vcmodule only)	Branch tag of a vcfile or vcmodule version.	Without "[]", returns a list of branch tags associated with the current vcfile or vcmodule version picked out by the connection. With "[]", returns True/False depending on whether the string provided is a valid branch tag for the current version of the vcfile or vcmodule.
branchparent.* (vcfile or vcmodule only)	Parent vcfile or vcmodule of the branch of the current vcfile or vcmodule.	Returns the parent file or module of the branch of the current vcfile or vcmodule. Works only with a "..". Applies the next keyword to the parent vcfile or vcmodule version of the branch of the current vcfile or vcmodule version picked out by the connection. Without a "..", returns an error.
branchversion[].* (vcfile or vcmodule only)	The different versions of the branch of the vcfile or vcmodule.	Returns the different versions of the current vcfile or vcmodule in the branch. With no "[]" and no "..", returns in order the ids of the versions of the branch of the current vcfile or vcmodule version. With "[]" but no "..", returns True/False. With a ".." and no "[]", applies the keyword that follows to each version of the branch of the current vcfile or vcmodule version. With a "[]" and a "..", applies the next keyword to the version identified by the string in the brackets treating it as a version id.
subbranch[].* (vcfile or vcmodule only)	All branches emerging from a vcfile or vcmodule version.	With no "[]" or "..", returns the ids of all branches emerging from the vcfile or vcmodule version. With "[]" and no "..", returns True/False depending on whether the string provided is a valid branch id or branch tag emerging from the current version of vcfile or vcmodule. With a ".." and no "[]", applies the next keyword to all the branches emerging from the current vcfile or vcmodule version picked out by the connection. With "[]" and "..", applies the next keyword to the last version of the branch whose branch id or branch tag is inside the brackets. The branch should be emerging from the current vcfile or vcmodule version picked out by the connection.

config.*	Identify the configuration of the vcfile or the folder containing the vcfile or vcmodule.	Returns the configuration name as provided in the configuration clause.
created (vcfile or vcmodule only)	Date and time the vcfile or vcmodule version was created.	Returns the date and time of the creation of the file or module.
created.generic (vcfile or vcmodule only)	Date and time in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Returns the date and time in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) of the creation of the file or module.
complete (vcfile or vcmodule only)	Is vcfile or vcmodule checkin complete?	True/False. For information on the [in]complete flag see Connect to a File in DesignSync .
checkintag (vcfile or vcmodule only)	A string containing the autotag (the version tag) that was assigned during checkin.	String Value
checkintag (vcfolder only)	A string containing the autotag (the version tag) that was assigned during checkin.	String Value
comment (vcfile or vcmodule only)	Comment entered by a user when checking in a vcfile or vcmodule.	Returns the checkin log entry for the current vcfile or vcmodule.
description	Description of the connection.	Returns the description as provided in the description clause.
format[].*	Format of the connection.	Used as "format", returns the name of the format of this vcfile or vcmodule. Used as "format[]", returns True/False. Used as "format.*", applies the next select keyword to the format of the connection.
index (vcfile or vcfolder only)	ENOVIA generated index number of the connection.	Returns the unique index number of the connection.
locked (vcfile or vcmodule only)	Is the current vcfile or vcmodule version locked?	True/False.
locker (vcfile or vcmodule only)	Name of the user who has locked the current vcfile or vcmodule version.	Returns the name of the user who locked the file or module.
otherconfigs[].* (vcfolder only)	Other configurations that are applicable to the vcfolder.	Returns information about other configurations applicable to the vcfolder. Used as otherconfigs, returns a list of all names of configurations of this vcfolder in DesignSync. Used as otherconfigs[], returns True if the string inside the brackets is the name of a configuration of the vcfolder. Used as otherconfigs[].*, applies the next keyword to this folder as if its configuration was given by the string in the bracket.
parent.* (vcfile or vcmodule only)	Parent vcfile or vcmodule of the current vcfile or vcmodule version.	Returns the parent file for the current vcfile or vcmodule. Works only with a "..". Applies the next keyword to the parent vcfile or vcmodule version of the current vcfile or vcmodule version picked out by the connection. Without a "..", returns an error.

path	Path of the connection.	Returns the path provided in the connect business object command.
qualifier (vcfile or vcmodule only)	Identify the version of the branch picked out by the vcfile or vcmodule connection (at any given point in time).	Returns the value as provided in the qualifier clause. If no value was entered, returns an empty string " ". The qualifier clause is applicable for spectypes branchid and branchtype only.
retired (vcfile or vcmodule only)	Is the branch of the current vcfile or vcmodule version retired?	True/False. Returns True if the branch of the file or module is retired. Otherwise returns False.
subfolder[].* (vcfolder only)	Vcfolders that are subfolders of the vcfolder.	Returns information on subfolders that are part of the vcfolder. Used as subfolder, returns the names of the vcfolders that are sub-folders of the vcfolder. Used as subfolder[], returns True/False. Used as subfolder[]., applies the next keyword to the sub-folder whose name is within the brackets. Used as subfolder.*., applies the next keyword to each vcfolder that is a sub-folder of the vcfolder picked out by the connection.
spectype (vcfile or vcmodule only)	Type of the specifier of the connection.	Returns one of the following types: versionid/versiontag/branchid/branchtag/selector.
specifier (vcfile or vcmodule only)	Specifier of the connection.	Returns the specifier of the connection as provided by the user.
store[].*	Store of the connection.	Used as "store", returns the name of the store of this vcfile or vcmodule. Used as "store[]", returns True/False. Used as "store.*.", applies the next select keyword to the store of the connection.

Selectables that return data held in the DesignSync system.

DesignSync data Selectable Fields

Field	Description	Output
vcname	Name of a vcfile, vcfolder, or vcmodule version.	Returns the name of the current vcfile, vcfolder, or vcmodule version picked out by the connection.
vcpath	Path of a vcfile, vcfolder, or vcmodule version.	Returns the path relative to the path in ENOVIA Live Collaboration store definition of the version of the vcfile, vcfolder, or vcmodule picked out by the connection. When used with vcfile, the path includes the file name of the vcfile.
versionid (vcfile or vcmodule only)	Id of a vcfile or vcmodule version.	Returns the unique id of the current file or module picked out by the connection.
versiontag[] (vcfile or vcmodule only)	Version tag of a vcfile or vcmodule version.	Without "[]", returns a list of version tags associated with the current vcfile or vcmodule version picked out by the connection. With "[]", returns True/False depending on whether the string provided is a valid version tag for the current version of the vcfile or vcmodule.
vcfile[].* (vcfolder only)	Vcfiles that are part of the vcfolder.	Returns information about vcfiles that are part of the vcfolder. Used as vcfile, returns the name of each vcfile. Used as vcfiles[] True/False depending on whether

		<p>the string inside the brackets is the name of a file in the folder.</p> <p>Used as <code>vcfile[].*</code>, applies the next keyword to the vcfile version indicated by the string within the brackets if it is the vcname of one of the files in the folder. The vcfile version is treated as if it were a vcfile connection identified by its version id.</p> <p>Used as <code>vcfile.*</code>, applies the next keyword to all the vcfile versions in the same as is done with brackets.</p>
<code>vcfolder.*</code> (vcfile or vcmodule only)	The vcfolder (DesignSync folder) of which the vcfile or vcmodule is a part of.	<p>Works only with a "..".</p> <p>Applies the next keyword to the vcfolder of this vcfile or vcmodule treated as a vcfolder connection with the config of this connection. If there is no config, the operation of certain keywords will vary depending on the store.</p>

Selectable Fields for Abstract Vcfile and Vcmodule

This table lists selectable fields for the abstract vcfile and vcmodule.

Selectable Fields for abstract vcfile		
Field	Description	Output
root	Provides information about the initial version of the vcfile or vcmodule tree of versions.	Without a ".", returns the vcpath. With a ".", applies the next keyword to it as if it were a vcfile or vc module connection identified by its version id.
version[].*	Identifies the version of the vcfile or vcmodule. It has the same format as VERSION of a vcfile or module connection. For information on how to specify the VERSION refer to Specify Version .	Include the version number in []. Without a ".", returns True/False. With a ".", applies the next keyword to the version identified.

History for DesignSync File Access

For all vcfolder, vcfolder, vcmodule operations, history is recorded in the ENOVIA Live Collaboration database.

The records include the vcconnection index. Therefore, the actions on a particular vcconnection can be followed throughout the lifetime of the business object to which it is connected. The history records do not distinguish between vcfolder, vcfolder, or vcmodule connections.

The following MQL statement returns the history for a vcconnection:

```
print bus TNR select history.vcconnection;
```

The history records for a vcconnection index are:

vc connect
vc modify connection
vc delete connection
vc remove current version
vc remove file
vc add tag
vc remove tag
vc lock
vc unlock
vc copy to store
vc checkin
vc checkout

Example:

In addition to the vcconnection index, the history records provide the following information:

vc connect
store, path, specifier, format, description
vc modify connection
store, path, specifier, format, description
vc add tag
tag added
vc remove tag
tag removed
vc copy to store
store copied to

The specifier output for a vcfolder connection is the configuration name. For a vcfolder or vcmodule connection, the output is the value provided in the VERSION string. For information on how to specify the VERSION, see [Specify Version](#).

Find the ENOVIA Object ID from DesignSync

When a business object is connected to a file or folder in a DesignSync server, the metadata about the business object and the connection is stored in DesignSync.

The metadata information consists of the business object ID in the ENOVIA Live Collaboration Server and the selector identifying the file or the folder version that is connected to the business object.

The syntax for printing the value of the business object ID stored in DesignSync is as follows:

```
Print bus TNR select | vcfile.businessobject |
| vcfolder.businessobject |
```

The business object ID is stored even when an empty connection is created. When a file or a folder is eventually checked in against this empty connection, DesignSync associates the saved business object ID with the version of the checked in file or configuration of the checked in folder.

Any command that results in the creation, deletion, or modification of connections between business objects and files or folders on a DesignSync server causes the metadata to be updated in the ENOVIA Live Collaboration Server as well as in DesignSync. The command completes successfully only if the operations involved are successful at both ends.

The metadata in DesignSync is affected and can change when a connection is established between a business object and a file or a folder in DesignSync or an existing connection is deleted or modified. Additionally, the metadata in DesignSync is updated during the following operations in ENOVIA products. In the following cases except where indicated, the system tries to perform the update after the transaction has finished. So any failure of the update does not affect the action performed in the ENOVIA product.

- When a business object is deleted, any references to it in DesignSync are removed.
- When a business object in a DesignSync store is cloned, the connections associated with the object are copied to the newly-created business object (if the default copy files option is used) and DesignSync metadata is updated to record information about the new business object.
- When a business object in a DesignSync store is revised, the connections are copied if the inherit files option is used and DesignSync metadata is updated to record information about the new business object.
- When a business object and its vcconnections are imported, information from imported connections is used to update DesignSync metadata about associations between DesignSync files or folders with the imported business object.

If the connections made during the import process fail, it causes the transaction to fail as well.

- When a business object ID is changed as a result of commands to change vault or cluster oids, DesignSync metadata is updated to replace the old ID of the business object with the new ID.

In all of the above operations, if the metadata information about the business object cannot be updated in DesignSync, a warning is issued but the operation itself does not fail.

Synchronize ENOVIA Updates with DesignSync

You can use the sync businessobjectlist command to synchronize one or more business objects in ENOVIA with DesignSync.

This MQL command executes a sync:

```
sync businessobjectlist BUS_OBJECT_COLLECTION_SPEC vcconnection;
```

BUS_OBJECT_COLLECTION_SPEC is one of:

```
set NAME  
query NAME  
temp set BO_NAME{,BO_NAME}  
temp query [TEMP_QUERY_ITEM {TEMP_QUERY_ITEM}]  
expand [EXPAND_ITEM {EXPAND_ITEM}]
```

You can also use these specifications in boolean combinations.

The keyword businessobjectlist can be shortened to buslist.

For example, the following command will synchronize all business objects inside the vault BusinessVault with the DesignSync store associated with the objects:

```
sync buslist temp query bus * * * vault BusinessVault where "vcfile==TRUE || vcfolder==TRUE" vcconnection;
```

Troubleshooting DSFA

You can troubleshoot some problems arising in your implementation of DesignSync File Access by enabling DS tracing.

ENOVIA recommends that you use tracing for DesignSync File Access interactively, using the following MQL commands:

```
trace type ds on;  
trace type ds off;
```

You can also enable tracing by adding MX_DS_TRACE = TRUE to your enovia.ini (Windows) or mxEnv.sh (Unix) file (or by setting it to a filename). The log shows the interaction between ENOVIA Live Collaboration and the DesignSync server.

Errors related to checkin or checkout are not logged.

PowerView

A PowerView page displays information from different locations in an application on a single page, allowing users to quickly access and update commonly-used information. The term portal refers to the administration object, not to the broader internet definition described in JSR 168.

In this section:

- [About PowerView Pages](#)
- [Building and Linking a PowerView Page](#)
- [Parameters and Settings for Portal Objects](#)
- [Parameters and Settings for Channel Objects](#)
- [Parameters for Tab Command Objects](#)
- [Settings for Tab Command Objects](#)
- [URL Parameters Accepted by emxPortal.jsp](#)

About PowerView Pages

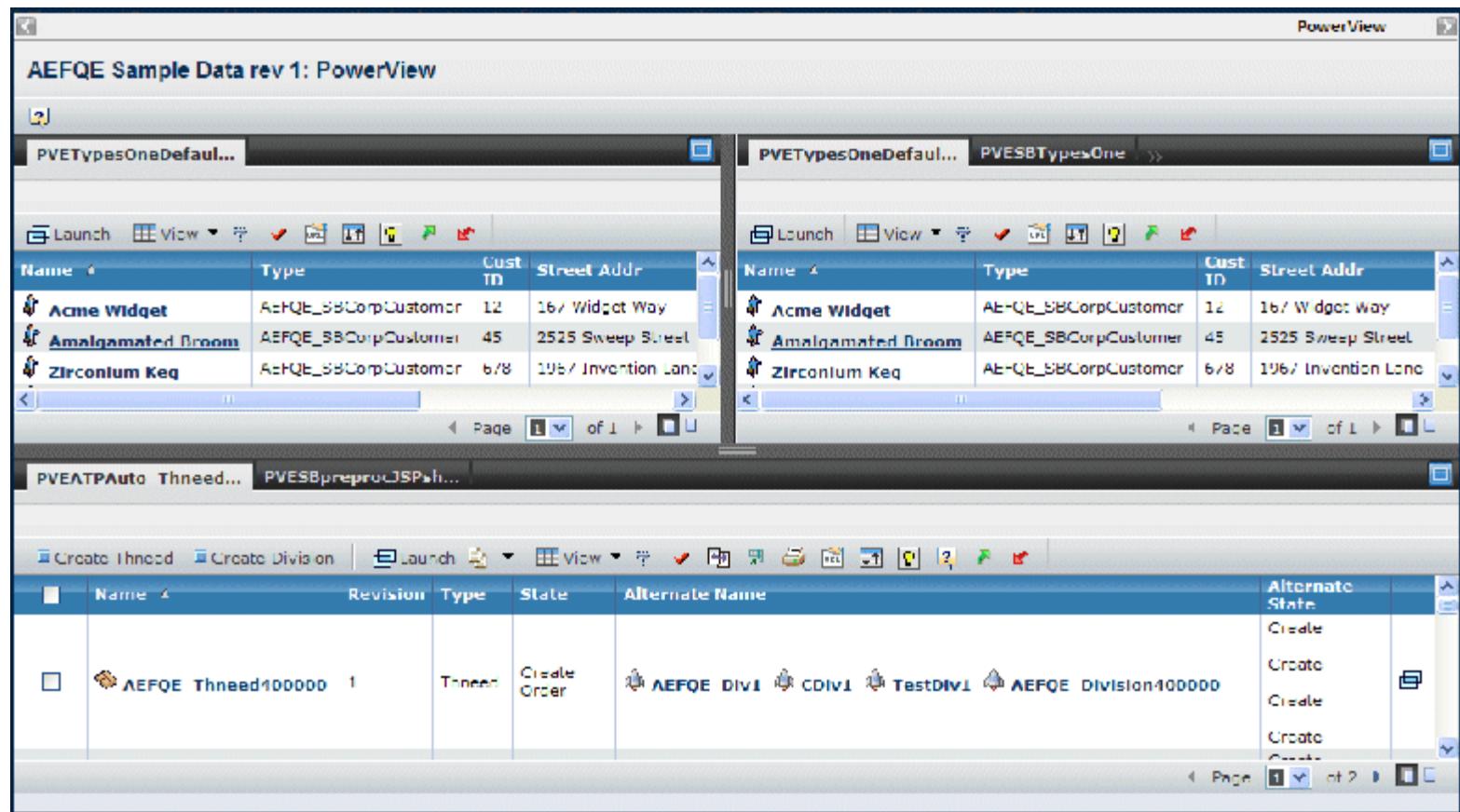
A PowerView can aggregate information about an object into a single page. The Business Administrator can set the PowerView as the default page to display when users initially access that object type.

The following topics are discussed:

- [PowerView Page](#)
- [PowerView Layout](#)
- [Channel Layout](#)
- [Tab Layout](#)
- [Application Link to the PowerView Page](#)
- [Portal Mode](#)
- [PowerView Launch Maximize](#)
- [Details Channel](#)

PowerView Page

The configurable JSP emxPortal.jsp creates PowerView pages. This JSP accepts parameters that define the content and behavior of the page. These parameters are passed to the JSP through the href URL.

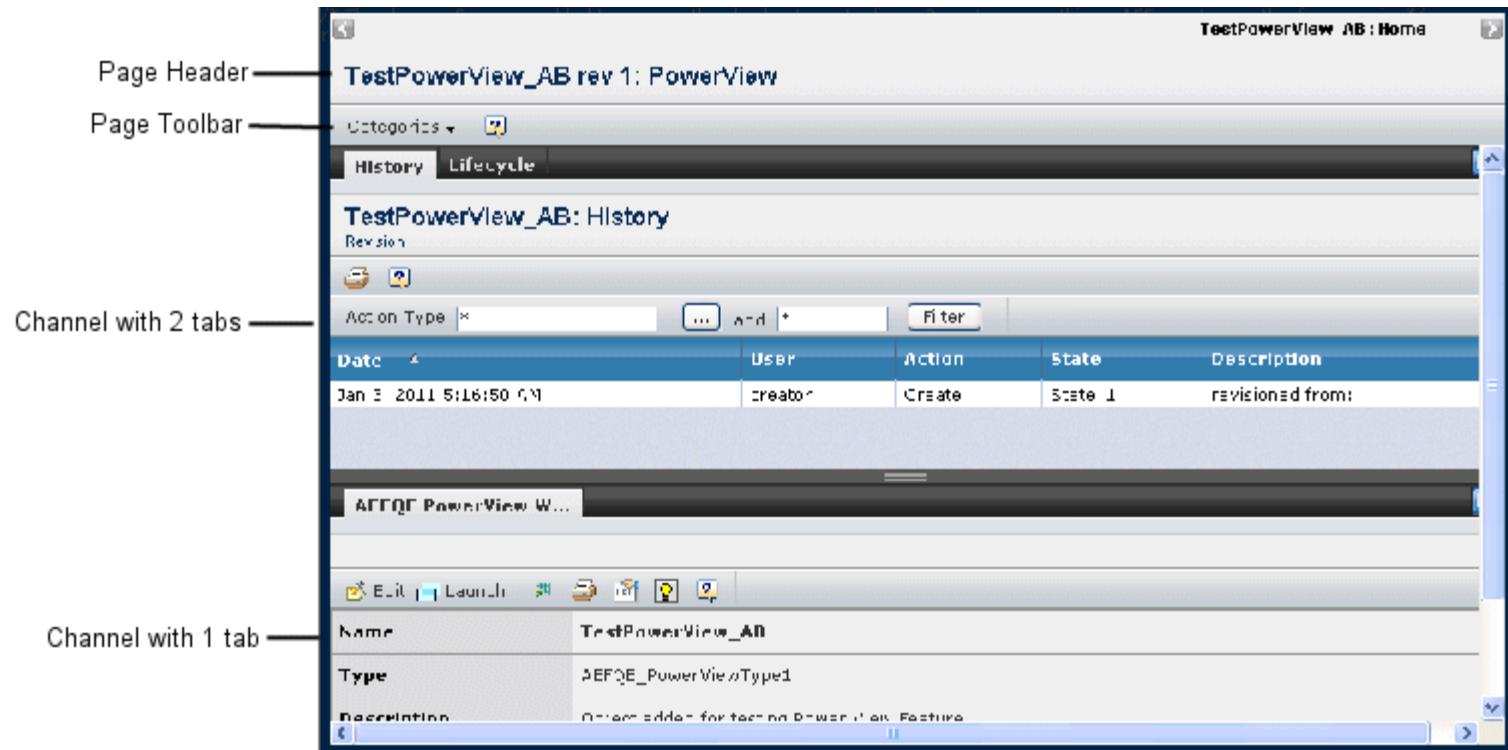


When the PowerView page is opened, each tab shows at its default size. Users can click the maximize button so that the tab takes up all the area within the PowerView page (hiding all other tabs); then the user can click the minimize button to return to the default size. The grabber between groups of tabs can be used to resize the channels as needed.

If a toolbar action within a channel targets a slide-in frame, then that channel is automatically maximized. When the slide-in window is closed, the user must minimize the channel.

PowerView Layout

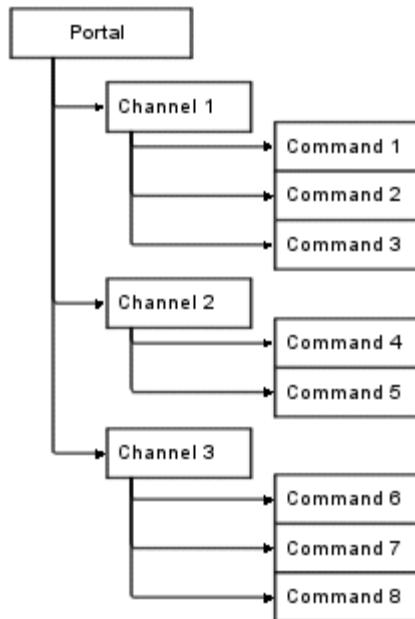
PowerView pages are composed of channels and tabs. This graphic shows the main components of a PowerView page.



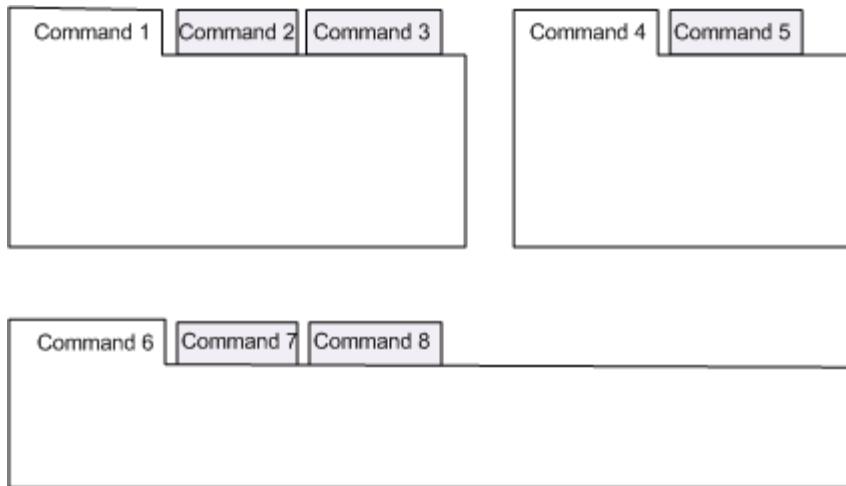
The channel tabs contain the content for users to work on or view. The tabs contain information typically displayed on an application page. Examples include: an Actions menu, a configurable table with pagination controls and filter list, a read-only or editable configurable form, a custom JSP, or any Web page.

The height of channels must be defined using percentages and the total of all channels must equal 100%. If not, emxPortal.jsp equally distributes the heights of the channels to equal 100%. the width cannot be specified; it is automatically equally distributed using the available area. If the browser window is resized, the heights and widths are resized proportionally.

This graphic shows the administrative objects required to define a PowerView page. The main components of the PowerView page are defined as portal, channel, and command administrative objects. The names of the top-level PowerView portal object and the toolbar menu object (not shown below) are passed to the emxPortal.jsp through its URL parameters.



The above administrative objects would create a PowerView page like the one shown below.



You need a portal administrative object for each unique PowerView page and one or more channel administrative objects for each unique portal. You can assign channel administrative objects to more than one portal object.

A form field or column can use the `Target Location = command name` setting, where the command name is a defined tab, so that when clicked, that specified tab is loaded into the PowerView.

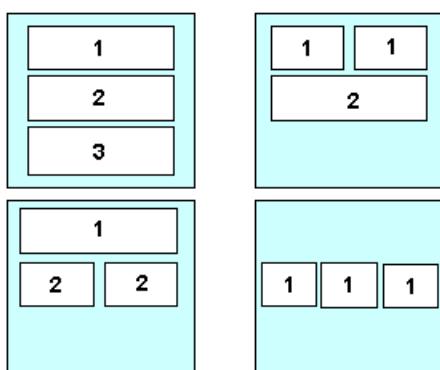


Channel Layout

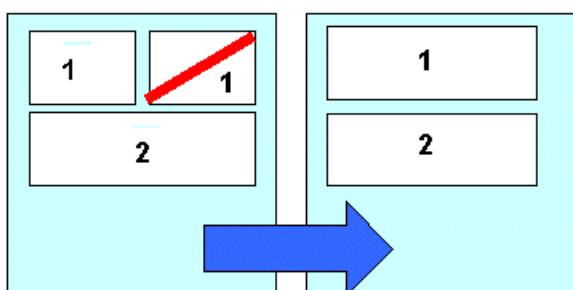
A channel can contain one or more tabs. There is no limit for the number of channels that you can display in a row or on a single PowerView page, but displaying more than 2 channels in a row or more than 3 rows can make the page difficult to use. You cannot add a multiple page wizard as a channel. The Portal component does not support multiple page commands.

The row a channel appears in is determined by how it is assigned to the portal object. If you are using Business Modeler to create portal objects, channels can be arranged in the Items tab to reflect how they will appear on the page. In MQL, use the Channel clause of the Create Portal or Modify Portal command to arrange the channels. See the *Business Modeler Guide* or *MQL Guide* for details.

For example, channels can appear one below the other, or they can be grouped to appear side by side.



The PowerView page automatically compensates if the user does not have access to any of the tabs in the channel by hiding the channel and rearranging the display of channels. For example, if the user does not have access to the second channel in the first row, as shown in the left of the figure below, the layout shown on the right displays instead.



Displaying table or structure browser components in half page-width channels typically causes the user to scroll horizontally to view all columns in the table, but you can create an alternate table with fewer columns for display specifically within a channel tab. See [URL Parameters Accepted by emxTable.jsp and emxTableEdit.jsp](#).

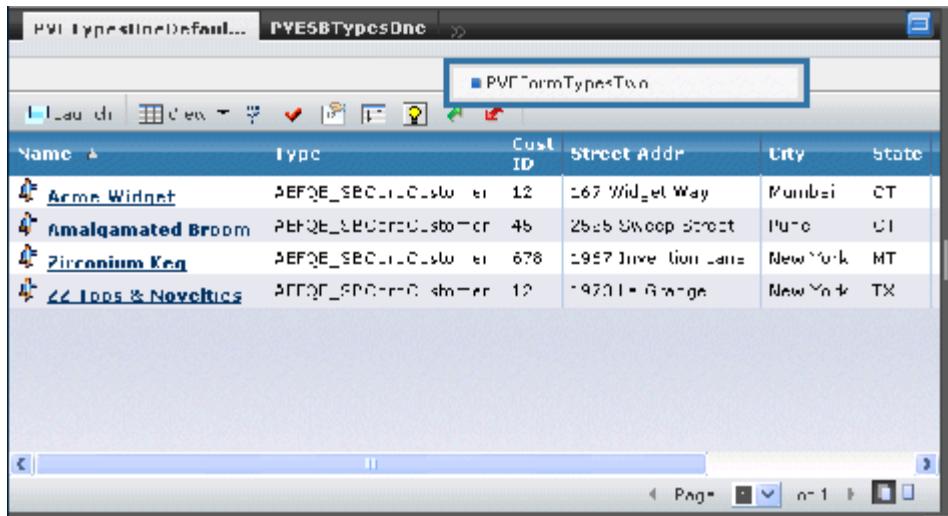


Tab Layout

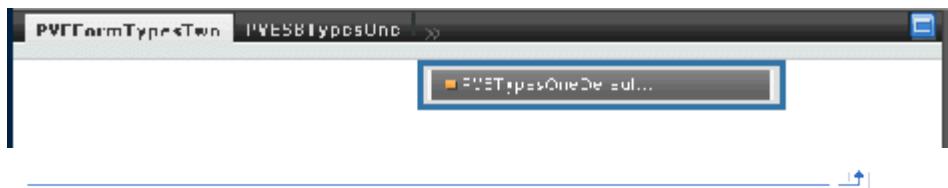
The tabs within channels contain the actual content to be viewed or worked on. The display order of the tabs within a channel is determined by the order you add the tab commands to the channel object.

You can display up to 4 tabs on a full, page-width channel and 2 tabs for half page-width channels. For channels with more tabs, the system displays a small arrow tab. This tab provides a drop-down menu of the remaining tabs. When the user selects one of the items in the drop-down menu, the system replaces the currently-selected tab with the newly-selected tab. The system puts the previous selection in the drop-down menu.

This graphic shows a channel that has 3 tabs. The 3rd tab is accessed using the >> drop-down menu. When the 3rd tab is selected...



...it replaces the previously-selected tab. The previously-selected tab is then listed in the >> drop-down menu.



Application Link to the PowerView Page

The PowerView component allows you to link it to an application as a desktop or object PowerView.

Business Process Services installs the portal administrative object called AEFPowerView, and the first channel of the Desktop PowerView page, called AEFPowerViewChannel. To add to that channel, use the command called AEFCollections. Each installed ENOVIA product can add channel objects to this object to define channels appropriate for that application. For example, Sourcing Central might add a channel that contains tabs for collections, RFQs, and Buyer Desks. You can configure the channels and tabs assigned to AEFPowerView as needed but this object must be used for the Desktop PowerView.

Users typically access the Desktop PowerView page by clicking > My Tools > PowerView from the global toolbar or by clicking the Home tool from the global toolbar (if the Home page preference is set to PowerView). The component that calls the Desktop PowerView page must specify the configurable JSP for PowerView pages, emxPortal.jsp, in the href parameter. The href must include a parameter that specifies AEFPowerView. Make sure the href includes any other parameters needed to configure the page, such as a toolbar menu object, as described in [Parameters and Settings for Portal Objects](#).

The order in which the channel objects are assigned to the portal object determines the order in which the channels display on the PowerView page. For example, the channels added for the first ENOVIA product installed listed first on the Desktop PowerView page, the channels for the second application installed are listed next and so on. To change the display order you must change the order in which they are assigned to the portal object.

For example, when entered in the href parameter for an appropriate command administrative object, the following URL creates a PowerView page:

```
 ${COMMON_DIR}/  
 emxPortal.jsp?portal=AEFPowerView&toolbar=AEFPowerViewToolbar&H  
 elpMarker=HelpMarker&header=PowerView
```

You can also link to an object PowerView.

Users typically access the PowerView page for an object type by clicking an object name from a table or by searching for a specific object. For example, in Engineering Central, clicking the name of an ECO from a table or search results displays the ECO PowerView page for that object.

The component that calls the object PowerView page must specify the configurable JSP for PowerView pages, emxPortal.jsp, in the href parameter. For example, when entered in the href parameter for an appropriate command administrative object, the following URL creates a PowerView page:

```
 ${COMMON_DIR}/  
emxPortal.jsp?portal=SPCSCOPortal&toolbar=SPCSCOToolbar&header=  
SummaryView&HelpMarker=emxhelpscosummary
```



Portal Mode

When users access the PowerView page, the system displays the data in portal mode where the page is divided into separate channels and each channel has separate display data associated with it.

Every page configured inside the PowerView includes the parameter `portalMode=true`, so that the page can read this request parameter and behave differently if required.

When a configurable table page is used in one of the PowerView channel tabs, the portal may display an alternate table instead of the regular table to conserve space. This alternate table can contain fewer columns than the regular table. You can pass the names of both the alternate table and the regular table as parameters when generating the PowerView page. The alternate table name is passed into the command href as `portaltable=PORTAL_TABLE_NAME`. So if you pass both the `portalMode` is true and the `portaltable` parameter, the channel displays using the alternate table.



PowerView Launch Maximize

In many cases, the tabs shown on the PowerView page correspond to pages opened using the Categories menu. You can configure the PowerView tab to conserve screen space so that it contains a subset of columns from the Category page.

The system provides a Launch button to launch the currently displayed channel tab in a maximized popup window that contains all the columns for the item. When a user clicks the Launch button, it passes the parameters `launched=true` and `portalMode=false` to the new window. The `portalMode` value must be read from the Request Object.

The Launch button is available in all the channels of the PowerView page regardless of whether the channel is a configurable table, configurable web form, or custom JSP.

When the user clicks on an object name hyperlink in the launched popup window, the page for that object opens in another popup window. Any edit operations performed in that page are reflected in the same popup after refresh. Neither the launched window nor the parent PowerView page are refreshed. As a general guideline, any update that happens in the application refreshes only the page in the first level.

In the normal summary page, if the user deletes an object, the system refreshes the summary page. In the portal view page, if the user deletes any object, the system refreshes only the summary page inside the channel. When the user deletes any object in the launched popup window, the system refreshes only that page. It does not refresh the PowerView page.

To force an update of the launched pop up window, you need to pass the Target Location setting on the command:

```
Target Location = formViewHidden
```

For example in the custom JSP that the command points to, use the following code to reload the page.

```
parent.document.location.href = parent.document.location.href
```



Details Channel

You can configure a master channel and a details channel. The tabs in the master channel could contain table or form pages that when you click a object defined with an href, the details channel is updated depending on the clicked object.

All the data in all tabs in the details channel pertain to the selected object in the master channel. In this example, when a user clicks a name in the Line Items tab, the tabs at the bottom of the PowerView are updated with the details for that line item.

The screenshot shows a SAP Fiori application titled "HAS RFQ 2: Line Item Power View". The main area displays a table of "Line Items" with columns: Name, Description, Required Quantity, Required U of M, and Received. A single row is selected, highlighted in pink, representing a "Fuel Tank X02" with a quantity of 1000 and a unit of measurement of ES (each). Below the table, a message indicates "1 object". At the bottom, there are tabs for "Attribute Group", "Attachments", "Supplier Inclusion...", and "TMA".

Fuel Tank X02: Attribute Groups

Name	Default	Role	Scope	State
Quantity	Yes	Supplier	Private - Specific Line Item	7/7
Req Attributes	Yes	Buyer	Public - All Line Item	7/7

To create master/details channels, you would (in addition to defining the actual contents (form or tables) for each tab and the steps in [Building and Linking a PowerView Page](#)):

- Define a column in a tab in the master channel with:
`href=emxRefreshChannel.jsp?channel=NameofDetailsChannel`
- Define the details channel with a single dynamic command.
- When the PowerView is initially loaded (no object selected in the master channel), the details channel is empty. On initial load, the refresh flag is not passed and the JPO does not return any dynamic commands without that flag.
- When object in master channel is clicked, pass the JPO method for the dynamic command with the objectId and the refresh flag. The JPO returns the required dynamic commands for the selected objectId.

Building and Linking a PowerView Page

This procedure explains how to create all the components needed for a PowerView page from scratch. You would use this procedure if you wanted to needed for a PowerView page from scratch. Use this procedure to create a PowerView page for an object type that does not have one by default. To configure the PowerView portal page or an existing object PowerView page, you add and remove portal, channel, menu and command objects and change settings for the objects.

1. In Business Modeler, create a portal object for the PowerView page.

2. Create a channel object for each channel that you want on the PowerView page.

When you create a channel object, you define whether it appears alone in a row on the page or in a row with other channels. For a description of how to fill in parameters and settings for channel objects, see [Parameters and Settings for Channel Objects](#).

3. Assign the channel objects to the portal object (created in Step 1).

You can drag the channel objects up and down or right and left within the Items tab to rearrange their display order in the PowerView page. For details, see [PowerView Layout](#).

4. Create a command object for each tab to include for each channel.

For a description of how to fill in parameters and settings for tab command objects, see [Parameters for Tab Command Objects](#) and [Settings for Tab Command Objects](#).

5. Assign the tab command objects to the channel that it belongs to.

The order you add the tab commands to the channel object determines the order they display on the channel. You can rearrange the Channel Items tab by dragging them up and down or left and right.

6. Make sure to create the administrative objects for the menus, toolbars, table pages, and form pages needed for the PowerView page and any of its channels/tabs.

7. In the href parameter for the menu or command object that calls the table page (or in a JSP calling the page), enter emxPortal.jsp and specify the parameters necessary to display the page. Since emxPortal.jsp is in the ematrix/common folder, the first part of the URL is typically: \${COMMON_DIR}/emxPortal.jsp. Here is an example of an href value for an object PowerView page.

```
${COMMON_DIR}/emxPortal.jsp?portal=SPCSCOPortal&toolbar=SPCSCOToolbar  
&header=SummaryView&HelpMarker=emxhelpscosummary
```

For a description of the parameters, see [URL Parameters Accepted by emxPortal.jsp](#).

8. To see the changes within the Web-based user interface, click  To see the changes within the Web-based user interface,

click  > Utilities > Reload Cache in the toolbar and click the browser Refresh button. The system automatically refreshes the cache when the component age expires. This setting is in emxSystem.properties.

Only the Administration Manager role have access to the Reload Cache tool.

Parameters and Settings for Portal Objects

This table describes the available parameters for portal, channel, and command objects that represent a PowerView page.

For specific instructions on how to create objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Name (specified in the channel Items tab in Business Modeler)	Name of the portal object. For naming conventions, see Naming Conventions for UI Administrative Objects .	--
Channel	The channels associated with the portal object	Names of channel objects, such as: AEFCollections AEFPowerViewChannel ENGECCChannel

Parameters and Settings for Channel Objects

This table describes the available parameters for channel objects that represent a PowerView page.

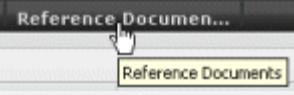
For specific instructions on how to create objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Name	Name of the channel.	Name of the channel
Height	The height of the channel as a percentage. The sum of all stacked channels should add up to 100%. If not, the PowerView will equally distribute the available height.	50%
Commands (specified in the channel Items tab in Business Modeler)	The command objects that represent the tabs in the channel. The order of the commands is the order the tabs appear in the channels in the user interface.	Names of command objects

Parameters for Tab Command Objects

This table describes the available parameters for command objects used for tabs that appear within channels on PowerView pages.

For specific instructions on how to create command objects using Business Modeler or MQL, refer to the *Business Modeler Guide* or *MQL Guide*.

Parameter	Description	Accepted Values/Examples
Access	<p>The persons, roles, and groups who can access the tab. To make the tab available to all users, regardless of role/group assignments, choose All.</p> <p>If no users are assigned access, the system assumes all users have access. If a user doesn't have access to any tabs in a channel, the channel is not displayed.</p> <p>Also see User Access to UI Components.</p>	Names of group, role, person administrative objects. <i>Or</i> All (default)
Alt	<p>The Alt text for a command is ignored. The Label, not the Alt value, displays when the mouse hovers over the tab. If the tab's label is truncated because it exceeds the limit, the full Tab label text is shown in the tooltip.</p>	WBSTasks emxEngineeringCentral.common.ECRs
Href	<p>The URL to display when a user selects the tab.</p> <p>Tabs can display content from the following sources:</p> <ul style="list-style-type: none"> configurable table configurable form JSP page URL <p>If the specified URL contains a header parameter, it is not used because the header would be redundant with the tab label. For example, if the URL contains emxTable.jsp with a header specified, the table header is not used.</p>	emxTable.jsp?xxxxx emxForm.jsp?xxxxx Sample.jsp?xxxxx http://www.matrixone.com
*Label	<p>The text to display on the tab. The label can be:</p> <p>Plain text, such as "ECRs".</p> <p>A string resource ID, such as "emxEngineeringCentral.common.portal".</p> <p>To internationalize the text, you must use a string resource ID. See Internationalizing Dynamic UI Components. The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text and/or output from the select expression.</p> <p>The label can be up to 17 characters long. If it exceeds 17 characters, the system truncates it to 17 characters and adds an ellipsis ("...") to the label. This default can be changed in emxsystem.properties with the property emxFramework.PowerView.Channel.Label.MaximumLength. The HTML "Title" attribute is defined so that when the user mouses over the tab, the full label displays.</p> 	Properties EBOM Lifecycle emxEngineeringCentral.common.portal
Settings	Additional settings that define the behavior and appearance of the tab command. For a list of the accepted settings, see the table below.	Name/value pairs, as defined in the table below.

Settings for Tab Command Objects

This table lists and describes the settings for command objects used for portal objects. Note that the name and value for each setting are case sensitive.

Setting	Description	Accepted Values/Examples
Access Expression	All these settings can be used to control access to tabs just as they can be used to control access to other UI components. For details, see User Access to UI Components .	--
Access Function		
Access Mask		
Access Program		
*Registered Suite	The application the command belongs to. The system looks for files related to the command in the registered directory for that application, which is specified in emxSystem.properties.	<p>Set the value with no spaces, for example, EngineeringCentral or Framework. Set the value to the suite name as defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. If the suite name starts with "eServiceSuite" then this prefix can be skipped and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite".</p> <p>In the href URL called when the tab is clicked, the system passes a parameter called "suiteKey". The value for the parameter is the property name from emxSystem.properties that maps to the setting's value.</p>

*Required Setting

URL Parameters Accepted by emxPortal.jsp

This table lists the parameters that emxPortal.jsp can use. You can add these parameters to the href parameter for the component that calls the portal JSP.

For example, when you specify the emxPortal.jsp to be called from a tree category, you can add these parameters to the href parameter for the menu object.

Parameter	Description	Accepted Input Values
Dynamic Command Function	Defines the method in the JPO specified by the Dynamic Command Program setting that returns a list containing the data structure to build the tabs in a details channel.	<JPO Method Name>
Dynamic Command Program	Defines the JPO invoked from a table cell or form field in a master channel to populate a details channel.	<JPO Name>
header	The content of the header to display at the top of the PowerView page.	Any alphanumeric text or a string resource ID. header=PowerView header="ECR Summary"
HelpMarker	Specifies the name of the help marker to call for context-sensitive help. For information about implementing help, see Implementing Context-Sensitive Help for FrameMaker Source .	String The naming convention for help markers is "emxhelp" followed by the object or feature and then the action, for example, emxhelproutecreate and emxhelpprojectedit. The marker is all lowercase with no spaces.
portal	Specifies the portal administrative object that represents the top-level menu, which defines the channels.	Name of portal object
showPageHeader	Enables the page header in the PowerView page. When set to false, the header(including the toolbar) does not display in the PowerView page.	true (default) false
showPageURLIcon	When true,  shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false, ICON does not show in the toolbar. The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in <code>emxSystem.properties</code> .	true false
subHeader	Creates a subHeader below the main header in the PowerView header frame.	The value can be any static text or a string resource id. The value can also include macros such as \$<type> \$<revision>.
TipPage	Specifies whether the page should include the Tip Page tool and call a specific html or JSP when a user clicks the tool. If not included, the Tip tool is not included on the page.	Name of a custom html or JSP page, including any path. The starting point for the directory reference is the content directory. For example, if you want to call an html file in ematrix/doc/customcentral and the content directory is ematrix/customcentral, you would add this parameter to the table.jsp: TipPage=../doc/customcentral/tippage.html
toolbar	Specifies the menu administrative object that represents the toolbar, which appears in the page header.	Name of menu administrative object.

JavaScript APIs

This section describes JavaScript APIs. For details on JPO APIs, see the Javadocs.

In this section:

- [emxEditableTable APIs](#)
- [Markup APIs](#)
- [Miscellaneous APIs](#)

emxEditableTable APIs

This section describes the APIs that can be used with emxEditableTable.jsp.

In this section:

- [Selecting Rows](#)
- [Expanding Rows](#)
- [Getting Cell Values by Row ID](#)
- [Getting Cell Values by Object ID](#)
- [Setting Cell Value by Row ID](#)
- [Setting Cell Values By Object / Relationship ID](#)
- [Enabling / Disabling Cell Edit by Row ID](#)
- [Enabling / Disabling Cell Edit by Object / Relationship ID](#)
- [Getting Current Cell Details](#)
- [Getting Parent ID](#)
- [Getting Child IDs](#)
- [Getting Child Column Values](#)
- [Getting Parent Column Values](#)
- [Reloading Cell Values](#)
- [Refreshing Table Columns](#)
- [Removing Selected Rows](#)
- [Complete Cell Information](#)
- [Displaying Validation Errors](#)

Selecting Rows

This API selects rows from the rows passed to the API. If a row Id does not exist, the API skips that row and does not throw an exception.

```
emxEditableTable.select
```

This API requires this input argument:

- arrRowIds. An array of row IDs.

Example:

```
Var arrRowIds = {[0,1],[0,0,2]}  
emxEditableTable.select(arrRowIds)
```

The example statement selects rows [0,1] and [0,0,2].

Expanding Rows

This API expands the rows to the specified level. If the structure does not have the number of levels, the API expands the entire structure. If a row Id does not exist, the API skips that row and does not throw an exception.

This API requires this input argument:

- arrRowIds. An array of row IDs.

Example:

```
Var arrRowIds = {[0,1],[0,0,2]};  
Var level =1;  
emxEditableTable.expand(arrRowIds)
```

The example expands the specified row up to 1 level.

Getting Cell Values by Row ID

This API returns `completeCellInfo` for the row ID and column name passed in.

See [Complete Cell Information](#).

`emxEditableTable.getCellValueByRowId`

This API requires these input arguments:

- `rowId`. The row ID of the needed cell.
- `colName`. The column name of the needed cell.

Example:

```
Var rowId = [0,1];
Var colName = FindNumber
emxEditableTable.getCellValueByRowId(rowId ,colName)
```

Getting Cell Values by Object ID

This API returns `completeCellInfo` for the row ID and column name passed in.

See [Complete Cell Information](#).

`emxEditableTable.getCellValueByObjectRelId`

This API requires these input arguments:

- `relId`. The relationship ID of the needed cell.
- `objectId`. The object ID of the needed object.
- `colName`. The column name of the needed cell.

Example:

```
Var relId = [];
Var objectId = [];
Var colName = FindNumber
emxEditableTable.getCellValueByObjectRelId(relId,objectId
, colName)
```

Setting Cell Value by Row ID

This API sets the actual and display values of the cell identified by the row ID and column name.

`emxEditableTable.setCellValueByRowId`

This API requires these input arguments:

- `rowId`. The row ID of the needed cell.
- `colName`. The column name of the needed cell.
- `cellValue`. The actual value of the cell to be stored in the database.
- `cellDisplayValue`. The value to be shown in the UI for the cell.
- `isRefreshView`. A boolean value that when true (the default), refreshes the structure browser page.

Example:

```
Var rowId = [0,1];
Var colName = Units
Var cellValue = U1,U2,U3;
Var cellDisplayValue = "U1 - U3";
Var isRefreshView = true;
emxEditableTable.setCellValueByRowId(rowId,colName,cellValue,
cellDisplayValue,isRefreshView)
```

This example sets the value for the Units column of row [0,1].

Setting Cell Values By Object / Relationship ID

This API sets the actual and display values of the cell identified by the row ID and column name.

```
emxEditableTable.setCellValueByObjectRelId
```

This API requires these input arguments:

- **relId**. The relationship ID of the needed cell.
- **objectId**. The object ID of the needed cell.
- **colName**. The column name of the needed cell.
- **cellValue**. The actual value of the cell to be stored in the database.
- **cellDisplayValue**. The value to be shown in the UI for the cell.
- **isRefreshView**. A boolean value that when true (the default), refreshes the structure browser page.

Example:

```
Var relId = [];
Var objectId = [];
Var colName = Units
Var cellValue = U1,U2,U3;
Var cellDisplayValue = "U1 - U3";
Var isRefreshView = true;
emxEditableTable.setCellValueByRowId(relId,objectId,colName,cel
lValue,cellDisplayValue,isRefreshView)
```

Enabling / Disabling Cell Edit by Row ID

This API enables or disables edit mode for a cell identified using the RowId and column name.

`emxEditableTable.setCellEditableByRowId`

This API requires these input arguments:

- `rowId`. The row ID of the needed cell.
- `colName`. The column name of the needed cell.
- `flag`. If true, sets the editMask attribute of the row to false (allows to row to be edited; if false, removes the editMask attribute from the row (the row cannot be edited).
- `isRefreshView`. A boolean value that when true (the default), refreshes the structure browser page.

Example:

```
Var rowId = [0,1];
Var colName = Units
Var isRefreshView = true;
emxEditableTable.setCellEditableByRowId(rowId,colName,true,isRefreshView)
```

In this example, the cell in the Units column of row [0,1] is made editable.

Enabling / Disabling Cell Edit by Object / Relationship ID

This API enables or disables edit mode for a row identified using the RelId, ObjectId, and column name.

```
emxEditableTable.setCellEditableByObjectRelId
```

This API requires these input arguments:

- `relId`. The relationship ID of the needed cell.
- `objectId`. The object ID of the needed cell.
- `colName`. The column name of the needed cell.
- `flag`. If true, sets the editMask attribute of the row to false (allows to row to be edited; if false, removes the editMask attribute from the row (the row cannot be edited).
- `isRefreshView`. A boolean value that when true (the default), refreshes the structure browser page.

Example:

```
Var relId = [];
Var objectId =[];
Var colName = Units
Var isRefreshView = true;
emxEditableTable.setCellEditableByObjectRelId(relId,objectId,
colName,true,isRefreshView)
```

In this example, row [0,1] is set to be editable.

Getting Current Cell Details

This API returns `completeCellInfo` for the cell currently in focus.

See [Complete Cell Information](#).

`emxEditableTable.getCurrentCell`

This API does not have any input arguments.

Example:

`emxEditableTable.getCurrentCell()`

Getting Parent ID

This API returns the row ID of the parent row for the current cell.

`emxEditableTable.getParentRowId`

This API requires this input argument:

- `rowId`. The row ID of the needed cell.

Example:

```
Var rowId = [0,1];
emxEditableTable.getParentRowId(rowId)
```

Getting Child IDs

This API returns an array of rowIDs of the first-level children for the current cell.

`emxEditableTable.getChildrenRowIds`

This API requires this input argument:

- `rowId`. The row ID of the needed cell.

Example:

```
Var rowId = [0,1];
emxEditableTable.getChildrenRowIds(rowId)
```

Getting Child Column Values

This API returns an array of column values of child cells based on the parent row ID and the specified column name.

```
emxEditableTable.getChildrenColumnValues
```

This API requires these input arguments:

- `rowId`. The row ID of the parent row.
- `colName`. The column name of the needed cell.
- `level`. The number of levels beneath that parent for which values should be returned.

Example:

```
Var rowId = [0,1];
Var colName = Units
Var level = 3;
emxEditableTable.getChildrenColumnValues(rowId,colName,level)
```

Getting Parent Column Values

This API returns the value of the parent object's cell based on the specified row ID, column name and level.

`emxEditableTable.getParentColumnValue`

This API requires these input arguments:

- `rowId`. The row ID of the child cell.
- `colName`. The column name of the needed cell.
- `level`. The number of levels above that child for which values should be returned.

Example:

```
Var rowId = [0,3];
Var colName = Units
Var level = 1;
emxEditableTable.getParentColumnValue (rowId,colName,level)
```

Reloading Cell Values

If the column has the Reload Program and Reload Function settings defined, this API calls that method to reload the cell value. This API is often invoked from the JavaScript functions defined by the `onFocusHandler` or `onChangeHandler` settings for the structure browser column. The reload program provides the cell values for the entire row.

This API only reloads data for the cell currently being modified.

`emxEditableTable.reloadCell`

This API requires this input argument:

- `colName`. The column name of the needed cell.

Example:

```
reloadCell("states")
```

This example invokes the `reloadCell("states")` from `onChangeHandler` of column "country". This function loads new range values for column "states".

Refreshing Table Columns

These APIs let you refresh all columns in the structure browser.

Use the first one to refresh with sorting; use the second one to refresh without sorting:

```
emxEditableTable.refreshStructure  
emxEditableTable.refreshStructureWithOutSort
```

When using the `refreshStructure` API, the structure browser is sorted based on the current values for sort parameters (`sortColumnName` and `sortDirection` URL parameters).

This API does not require any input arguments.

Example:

```
emxEditableTable.refreshStructure()
```

Removing Selected Rows

This API removes rows from the view (the data is not deleted).

`emxEditableTable.removeRowsSelected`

This API does not require any input arguments.

Example:

`emxEditableTable.removeRowsSelected()`

Complete Cell Information

For some APIs, the method returns `CompleteCellInfo`.

The information contains this data:

```
Object.rowID  
Object.columnName  
Object.relid  
Object.objectid  
Object.type  
Object.level  
Object.direction  
Object.value.current.actual  
Object.value.current.display  
Object.value.old.actual  
Object.value.old.display  
Object.editable
```

The `object.direction` is with respect to the object's parent.

Displaying Validation Errors

This API displays validation errors from a custom JSP page. If a row Id does not exist, the API skips that row and does not throw an exception.

```
emxEditableTable.displayValidationMessags( "XML" );
```

This API requires this XML as the input argument:

```
<mxRoot>
<object rowId="0,5">
<error>Error: Object with the same name already exists.</error>
</object>
<object rowId="0,3,2">
<error>Error: Multiple connections are not permitted.</error>
</object>
</mxRoot>
```

The rowIDs and the text of the error messages, must be defined by the JSP.

Markup APIs

The APIs in this section add or remove rows to the structure browser.

In this section:

-  [Load Mark Up](#)
-  [Adding Rows to Selected Rows](#)

Load Mark Up

This API adds a new row to a structure browser. Applications must pass the object ID of the row to which the new row will be connected.

Within an XML <mxRoot> element, the markup value indicates the action to take, and can be any of these values:

- **add**. Adds a row for the object.
- **cut**. Removes the row for the object from the structure browser.
- **.** Updates column values.
- **.** Resequences rows.

This example adds the specified object as a child to the first object:

```
<mxRoot>
  <object objectId="19940.22650.58300.63486">
    <object objectId="19940.22650.28544.49549" relId=""
      relType="relationship_EBOM" markup="add"> </object>
    </object>
  </mxRoot>
```

This example removes the specified object:

```
<mxRoot>
  <object objectId="19940.22650.58300.63486" rowid="0">
    <object objectId="19940.22650.28544.49549"
      relId="19940.22650.44544.64603"
      parentId="19940.22650.58300.63486"
      relType="relationship_EBOM" markup="cut"/>
  </object>
</mxRoot>
```

Adding Rows to Selected Rows

The `addToSelected` API allows you to paste a row either above or below the selected row.

The `<data status>` element includes the `pasteBelowOrAbove="true"` attribute/ value, and the `<item>` element includes the data about the object to be pasted including the `pasteAboveToRow` or `pasteBelowToRow` attribute with the row data.

For example:

```
<mxRoot>
<action>
<![CDATA[refresh]]>
</action>
<data status="committed"  pasteBelowOrAbove="true">
<item oid="30536.59499.40784.21329"
relId="30536.59499.40784.14515" pid="30536.59499.60784.65279"
pasteAboveToRow="0,0" />
</data>
</mxRoot>
```

The pasted row can be set as editable or non-editable by adding the `RowEditable` (values of true or false) attribute to the `<item>` element as shown here:

```
<mxRoot>
<action>
<![CDATA[refresh]]>
</action>
<data status=" committed "  pasteBelowOrAbove="true">
<item RowEditable="false" oid="30536.59499.40784.21329"
relId="30536.59499.40784.14515" pid="30536.59499.60784.65279"
pasteBelowToRow="0,0" />
</data>
</mxRoot>
</mxRoot>
```

You can paste a row as the child of the selected row in committed or pending mode (defined by the data status element):

```
<mxRoot>
<action>add</action>
<data status='pending' fromRMB='false'>
<item oid='43472.32595.47845.16772'
relType='relationship_SBOdersToLineItems' relId=''
pid='40712.19048.41410.33' direction='from'>
<column name='Description'>Description 00</column>
</item>
</data>
</mxRoot>
```

Miscellaneous APIs

This section describes miscellaneous APIs.

In this section:

-  [Load Custom Styles](#)
-  [Disabling Check Boxes](#)

Load Custom Styles

This method loads custom styles (CSS class names) for the window from the program `dsecUITypeCustom.css`. Call this method in the html page to load the needed CSS style.

`loadCustomStyle`

This API does not have any input arguments.

This function does not work in Google Chrome.

Disabling Check Boxes

This API lets you disable a check box in the structure browser.

To disable a check box in the structure browser, send this key value pair:

`disableSelection = true|false`

Techniques

This section provides techniques for working with configurable components and custom applications.

In this section:

- ❑ [Procedure for Creating an Attribute](#)
- ❑ [Procedure for Adding a Subtype](#)
- ❑ [Procedure for Adding an Attribute to a Type](#)
- ❑ [Procedure to Add a Trigger](#)

Procedure for Creating an Attribute

Attributes can be used with Types and Relationships. When creating a new application or customizing an existing product, you can create new attributes to meet your business needs. You can use Business Modeler or MQL; this procedure describes how to create an attribute using Business Modeler.

See the *Schema Reference Guide* for a list of attributes, and for lists of the attributes used for specific types and relationships. For more details on all the options available when defining attributes, see the *Business Modeler Guide*.

1. Determine if an existing attribute meets your needs. If not, continue with this procedure.

Note: If you use an existing attribute, do not change its details (data type, range values, and so on). If you do, you could introduce problems to any product that uses that attribute.

2. Select **Object > New > Attribute**.

3. On the **Basics** tab, enter these details:

- Name. The attribute name must be unique within the ENOVIA database.
- Description. Describe the purpose of this attribute: how it will be used and the meaning of the data it holds.
- Type. Select the data type this attribute will store:
 - String
 - Real
 - Integer
 - Date and Time
 - Boolean
- Default. If the attribute should be pre-populated with a default value when the type or relationship is created, enter that value here.
- Dimension. For an attribute with the integer or numeric data type, select a dimension. Dimensions associate units of measurement with the attribute and are used for converting from one units to another (such as from one weight system to another).
- Max length. For an attribute with the string data type, enter the maximum number of characters that can be entered for the attribute value. The default maximum is 0, meaning that an unlimited number of characters can be entered for the attribute value.

4. Optionally, add an icon to display for this attribute. Only Business Administrators using Business Modeler see the icon; it is not used in any application that uses the attribute.

5. Select the check boxes you want to use for this attribute:

- Hidden. When checked, the attribute does not show in any attribute chooser lists (although it can be specified using MQL). This option can be used to hold values defined by software code, but not entered or viewed by the end user.
- Multiline. Available when the data type is set to string. When checked, text wraps to the next line as the user types and is shown in a scrollable text box. When not checked, the data entry field shows as a single line.
- Reset On. Works with the Clone and Revision checkboxes. When a type that uses this attribute is cloned or revised, this checkbox determines if the value for this attribute is retained (not checked) or reset to the default value (checked). If no default value has been provided, this checkbox is unavailable.
- Clone. Check if you want to use the default value for the attribute whenever a type that uses the attribute is cloned. Otherwise, the cloned type uses the current value of the original type.
- Revision. Check if you want to use the default value for the attribute whenever a type that uses the attribute is revised. Otherwise, the revised type uses the current value of the original type.

6. Define ranges for attributes where the user will select from a list of pre-defined values:

- a. Click the **Ranges** tab.
- b. Click **Add**.
The Add Range dialog box opens. The content of this box is customized for the attribute's data type.
- c. Enter the range value and click **OK**.
- d. Repeat steps b-c for each range value you want to add.

7. Add event triggers for the attribute. See [Triggers](#) for details on creating trigger programs.

- a. Click the **Trigger** tab.
- b. Click **Add**.
- c. Select the needed trigger. You can use the Name box and **Filter** button to search for the needed trigger.
- d. Click **OK**.
- e. Double-click the trigger name. The Edit Trigger dialog opens.
- f. Type the name of the program to associate with the trigger (or use the Program Chooser to locate it) in the text box corresponding to the type of trigger:

- Check. The trigger executes before the event occurs.
- Override. The trigger replaces the even transaction.
- Action. The trigger executes after the event occurs.

- In the Input field for each trigger, optionally define arguments to pass into the program.
- Repeat steps f-g for each trigger program.

8. Click Create.

9. Register the attribute:

See the *Application Exchange Framework User's Guide* for complete details on using the Utilities menu options.

- Log into the ENOVIA system as a user with the Administration Manager role.
- From the global toolbar, select  > Utilities > Property Registration > Admin Type.
- From the Admin Type drop-down list, select Attribute.
- In the Un-Registered Admins list, click the name of the new attribute.
- Enter values for the properties as needed. If you are using Business Process Services's convention for symbolic names, use the format [administrative type]_[object name with no spaces]:

- When entering the version number, use dashes as separators, not periods.
- When entering symbolic names for custom administrative objects, do not use the exact same naming convention as Business Process Services uses unless you are working closely with ENOVIA Engineering. You can use a slight variation of the naming convention, such as adding an abbreviation for your company's name. For example, instead of attribute_Color, a company with the abbreviation ACM could add attribute_ACMColor.

f. Click Create Registration.

The property values are saved and the type is moved into the Registered Admins list.

g. Click Close.

The attribute can now be added to types and relationships.

Procedure for Adding a Subtype

A subtype is derived from an existing type in a parent-child relationship, and then customized as needed. When defining the subtype (the child), you only need to specify the differences between it and the type you derived it from (the parent). A child inherits all attributes, methods, triggers, governing policies, and allowed types for relationships.

You can use Business Modeler or MQL; this procedure describes how to add a subtype using Business Modeler. See the *Schema Reference Guide* for a list of types. For more details on all the options available when defining types, see the *Business Modeler Guide*.

1. Determine which existing type is closest to your needs.

2. Select **Object > New > Type**.

3. On the **Basics** tab, enter these details:

- Name. Enter a name. The name must be unique within the ENOVIA system.
- Description. Describe the purpose of this type: how it will be used.
- Derived From. Click the ellipsis button and select the existing Type from Step 1.
- Abstract Type. Only check this box if this type will be used as a parent type in a hierarchy and not used to create business objects. The convention for naming abstract types is to use all UPPER CASE letters.
- Hidden. When checked, the type does not show in any type chooser lists (although it can be specified using MQL). This option can be used for types needed by software code, but not viewed by the end user.

4. Optionally, add an icon to display for this type.

5. Assign attributes to the subtype:

- a. Click the **Attributes** tab.
The dialog box lists the attributes inherited from the parent type.
- b. Click **Add**.
- c. Select the needed attributes. You can use the Name box and **Filter** button to search for needed attributes.
- d. Click **OK**.
- e. To remove any unneeded attributes: click the attribute name, then click **Remove**.

6. Assign even triggers to the subtype:

- a. Click the **Triggers** tab.
The dialog box lists the triggers inherited from the parent type.
- b. To remove any unneeded triggers: click the trigger name, then click **Remove**.
- c. Click **Add**.
- d. Select the needed trigger. You can use the Name box and **Filter** button to search for the needed trigger.
- e. Click **OK**.
- f. Double-click the trigger name. The Edit Trigger dialog opens.
- g. Type the name of the program to associate with the trigger (or use the Program Chooser to locate it) in the text box corresponding to the type of trigger:
 - Check. The trigger executes before the event occurs.
 - Override. The trigger replaces the even transaction.
 - Action. The trigger executes after the event occurs.
- h. In the Input field for each trigger, optionally define arguments to pass into the program.
- i. Repeat steps f-g for each trigger program.
- j. To remove any unneeded trigger: click the trigger name, then click **Remove**.

7. Assign methods to the type.

A method is a program that is associated with a type. These programs can be executed by users when they select any business object of this type. Programs selected as methods require a business object as a starting point for executing.

- a. Click the **Methods** tab.
- b. Click **Add**.
- c. Select the needed method. You can use the Name box and **Filter** button to search for the needed method.
- d. To remove any unneeded method: click the method name, then click **Remove**.

8. Register the subtype:

See the *Application Exchange Framework User's Guide* for complete details on using the Utilities menu options.

- a. Log into the ENOVIA system as a user with the Administration Manager role.
- b. From the global toolbar, select  > Utilities > Property Registration > Admin Type.

- c. From the Admin Type drop-down list, select Type
- d. In the Un-Registered Admins list, click the name of the new subtype.
- e. Enter values for the properties as needed. If you are using Business Process Services's convention for symbolic names, use the format [administrative type]_[object name with no spaces]:
 - When entering the version number, use dashes as separators, not periods.
 - When entering symbolic names for custom administrative objects, do not use the exact same naming convention as Business Process Services uses unless you are working closely with ENOVIA Engineering. You can use a slight variation of the naming convention, such as adding an abbreviation for your company's name. For example, instead of type_Part, a company with the abbreviation ACM could add type_ACMPart.
- f. Click **Create Registration**.
The property values are saved and the type is moved into the Registered Admins list.
- g. Click **Close**.

Procedure for Adding an Attribute to a Type

This procedure shows you how to add an existing attribute to a type, including adding the attribute to forms and tables that support the type. You can use Business Modeler or MQL; this procedure describes how to create an attribute using Business Modeler.

The stages in this procedure are as follows:

- [Add the Attribute to the Type](#)
- [Add the Attribute as a Field on Forms for the Type](#)
- [Add the Attribute as a Table Column for the Type](#)

Add the Attribute to the Type

1. Select **Object > Find**.
The Find Objects dialog box opens.
2. Select **Type** from the **Object** pull-down list.
3. Enter the name of the needed type. You can use the * wildcard.
4. Click **Find**.
5. Click the needed type to select it, then select **Object > Open > Edit**.

The Edit Type dialog box opens.

6. Click the **Attributes** tab.
7. Click **Add**.
8. Select the needed attribute. You can use the Name box and **Filter** button to search for it.
9. Click **OK**.

10. Click **Edit**.

The attribute has been added to the Type.



Add the Attribute as a Field on Forms for the Type

See the *Business Modeler Guide* for complete details on defining fields for a web form.

1. Open the needed web form:
 - a. Select **Object > Find**.
The Find Objects dialog box opens.
 - b. Select **Web Form** from the **Object** pull-down list.
 - c. Enter the name of the needed web form. You can use the * wildcard.
 - d. Click **Find**.
 - e. Click the needed Web Form to select it, then select **Object > Open > Edit**.
The Edit Web Form dialog box opens.
2. Click **Add**.
The Add Field dialog box opens showing the Expression tab.
3. On the Expression tab:
 - a. Click the ellipsis button for the Expression field. The Select Pattern dialog box opens.
 - b. Click the **Attribute** check box and click **Filter**.
 - c. Click **OK**.
 - d. Select the **Business Objects** option button.
 - e. If you want to define a label for the attribute other than the Name (to be entered on the Basic tab), check the **Custom Label** check box and enter the needed **Label**.
4. Enter basic details:
 - a. Click the **Basics** tab.
 - b. Enter the **Name** for the field.
The name you enter displays on the form unless you entered a Custom Label in step 6e.
 - c. Enter a **Description** for the field.
5. Enter link details:
If the field will not be clickable by the end user, skip this step.
 - a. Click the **Link** tab.

- b. Enter the **Href**, the URL for the page that opens when the user clicks the field.
- c. Enter the **Alt** text that displays when the user hovers the mouse over the field.
- d. If you have defined a JSP to retrieve a range of values to populate the field with, enter the URL for that page in the **RangeHref** box.
- e. Enter the **UpdateURL**, the URL for the page to display after the field is updated.

6. Enter settings for the field:

- a. Click the **Settings** tab.

Typical settings used to define a field include (see [Form Fields](#) for more info):

- Registered Suite. The application the column belongs to.
- format. How the data in the column should be displayed.
- Field Size. How long, in pixels, to make the field.
- Field Type. How the data for the field is populated.
- Input Type. For edit mode, how the user enters data.
- Access Expression, Access Function, Access Mask, Access Program. Define who has access to the column. See [User Access to UI Components](#).
- Editable. For edit mode, defines if the user can edit the field

- b. For each setting, enter the **Name** and **Value**, then click **Set**. See [Settings for Fields in Web Form Objects](#) for details about the available settings.

7. Define who has access to the field:

If you want this field accessible to all users who have access to the form, skip this step.

- a. Click the **Access** tab.

Avoid adding specific users. Typically, use this tab to grant access based on a person's role to the field. For more details on defining access to UI components, see [User Access to UI Components](#).

- b. Click **Add**.

- c. Select the needed role. You can use the **Name** box and **Filter** button to search for the needed roles.

- d. Click **OK**.

- e. Click **User [All]** then click **Remove**.

8. Click **OK** to save and close the Add Field dialog box.

9. In the Edit Web Form dialog box, place the field where you want it to appear in the form by clicking the field, then clicking **Move Up** or **Move Down** until it shows in the needed location.

10. Click **Edit** to save the web form.

The attribute shows as a field on the view and edit web form pages.



Add the Attribute as a Table Column for the Type

See [Table Columns](#) for complete details for defining columns in a table.

1. Open the needed table:

- a. Select **Object > Find**.

The Find Objects dialog box opens.

- b. Select Table from the **Object** pull-down list.

- c. Enter the name of the needed table. You can use the * wildcard.

- d. Click **Find**.

- e. Click the needed table to select it, then select **Object > Open > Edit**.

The Edit Table dialog box opens.

2. Click **Add**.

The Add Column dialog box opens showing the Expression tab.

3. On the Expression tab:

- a. Click the ellipsis button for the Expression field. The Select Pattern dialog box opens.

- b. Click the **Attribute** check box and click **Filter**.

- c. Click **OK**.

- d. Select the **Business Objects** option button.

- e. If you want to define a label for the column other than the Name (to be entered on the Basic tab), check the **Custom Heading** check box and enter the needed **Heading**.

4. Enter basic details:

- a. Click the **Basics** tab.

- b. Enter the **Name** for the column

The name you enter displays on the form unless you entered a Custom Label in step 6e.

- c. Enter a **Description** for the column.

5. Enter link details:

If the cells in the column will not be clickable by the end user, skip this step.

- a. Click the **Link** tab.
- b. Enter the **Href**, the URL for the page that opens when the user clicks a cell in the column
- c. Enter the **Alt** text that displays when the user hovers the mouse over the column

6. Enter settings for the column:

- a. Click the **Settings** tab.
- b. For each setting, enter the **Name** and **Value**, then click **Set**. See [Settings for Table Column Objects](#) for details about the available settings.

Typical settings that should be defined include (see [Table Columns](#)):

- Registered Suite. The application the column belongs to.
- format. How the data in the column should be displayed.
- Input Type. For edit mode, how the user enters data.
- Access Expression, Access Function, Access Mask, Access Program. Define who has access to the column. See [User Access to UI Components](#).
- Editable. For edit mode, defines if the user can edit the cell value.

7. Define who has access to the column:

If you want this column accessible to all users who have access to the table, skip this step.

- a. Click the **Access** tab.
Avoid adding specific users. Typically, use this tab to grant access based on a person's role to the field. For more details on defining access to UI components, see [User Access to UI Components](#).
- b. Click **Add**.
- c. Select the needed role. You can use the Name box and **Filter** button to search for the needed roles.
- d. Click **OK**.
- e. Click User [All] then click **Remove**.

8. Click OK to save and close the Add Column dialog box.

9. In the Edit Table dialog box, place the field where you want it to appear in the form by clicking the field, then clicking **Move Left or **Move Right** until it shows in the needed location.**

10. Click **Edit to save the table.**

Procedure to Add a Trigger

You can add additional triggers to a type to perform checks or actions as needed. You can use Business Modeler or MQL; this procedure describes how to create a trigger using Business Modeler.

Before you begin: See the Business Modeler Guide for complete details on creating a program.

1. Select **Object > New > Program**.

The New Program dialog box opens.

2. On the **Basics** tab, enter these details:

a. Enter a **Name**. The program name must be unique.

b. Enter a **Description**.

c. Click the ellipsis button to associate a **User** with the program.

The access available for the selected user is granted to the program when it runs, regardless of which user actually runs the program.

d. Indicate the type of code this program object executes:

- Java, when the code is a JPO (Java programming object).

- MQL

- External

e. Select an option for **Execute**:

- Immediate. The program runs within the current transaction.

- Deferred. The program is cued to run after the outer-most transaction is successfully committed.

f. If the program requires an existing business object, such as when promoting, check **Needs Business Object**.

g. For MQL or External programs that need to be executed on a client, and not within the web product, check **Downloadable**.

h. For External programs that can be piped to a program, check **Piped**.

i. For programs written in Tcl code, check **Pooled** to allow allocating a pool of Tcl interpreters without requiring the system to initialize, allocate and close the Tcl interpreter each time a Tcl program is run.

j. If you do not want the program to display in program choosers (the program is not designed to run on its own), check **Hidden**. The program can still be accessed using MQL.

3. Click the **Code** tab.

4. Write the code using one of these methods:

- Enter the code in the Code text box.

- Use an integrated development environment to write the code.

- Use a text editor.

5. If you entered the code in the Code tab, click **Compile**.

6. Click **Create**.

7. Create an eService Trigger Program Parameters object to call this trigger. See [Adding a Trigger for a Trigger Event](#).

Best Practices

This section provides guidelines to help you develop ENOVIA products. It is not a comprehensive list, but includes information about common errors and recommended development patterns. Each guideline identifies the coding environment it applies to and is organized alphabetically by category.

In this section:

- ❑ [Business Objects](#)
- ❑ [Code Cleanup](#)
- ❑ [Database Access](#)
- ❑ [Documentation](#)
- ❑ [General](#)
- ❑ [HTML Tags](#)
- ❑ [i18n](#)
- ❑ [Izdate](#)
- ❑ [JavaScript](#)
- ❑ [JSP](#)
- ❑ [Memory Leaks \(IE\)](#)
- ❑ [Miscellaneous](#)
- ❑ [MQL Syntax](#)
- ❑ [Objects](#)
- ❑ [Standard Pages](#)
- ❑ [Statements](#)
- ❑ [StringBuffer](#)
- ❑ [Stylesheets](#)
- ❑ [URLs](#)
- ❑ [Variables](#)
- ❑ [WebSphere](#)

Business Objects

This concept describes best practices for working with business objects.

The following topics are discussed:

- [Business Objects and the getAttributes\(\) Method](#)
- [Use Business Object IDs](#)

Business Objects and the getAttributes() Method

The Bo.getAttributes() returns an empty list unless you make this call first:

```
BusinessObjectAttributes list2 = bo.getAttributes(_context);
```

Your code should look like this:

```
BusinessObjectAttributes list2 = bo.getAttributes(_context); //  
new call  
AttributeList list = list2.getAttributes();  
AttributeItr itr = new AttributeItr(list);
```



Use Business Object IDs

Beginning with Studio Customization Toolkit version 9.0, open a business object by specifying its ID only. The routine returns type, name, revision, and vault.

Code Cleanup

This concept presents best practices related to code cleanup.

The following topics are discussed:

- [JavaScript Functions](#)
- [Unused Variables](#)

JavaScript Functions

A JSP should not contain JavaScript code. Remove unnecessary JavaScript functions from a page; delete them or move them to their own .js include file.

Applies to: JSP



Unused Variables

Remove unnecessary or unused variables. Do not create multiple variables for the same thing. If there is already a variable that contains the information needed, use it. Review Strings, Variables, and Object Instances.

Applies to: JSP, Bean, JPO

Database Access

This concept presents best practices for working with database access.

The following topics are discussed:

- [Cache Results](#)
- [Deadlocks](#)
- [Retrieve All Required Data for a Page at Once](#)
- [Retrieve Data for All Table Rows in One Call \(Program and ProgramHTML Columns\)](#)
- [Transaction Boundaries](#)
- [Transactions and Nesting](#)
- [Transactions and Integrations](#)

Cache Results

Consider writing data to a cache when the data is not likely to change often and if many users frequently use it. For example, if you obtain a list of first and last names and the names do not change often, create a person bean that caches first and last name against the username key in the application layer. Once a person retrieves the first and last name from the database, all other users will retrieve that information from the cache, saving a database trip.

Applies to: JSP, Bean, JPO



Deadlocks

A deadlock occurs when two tasks try to gain control of the same object and each task has a lock on a resource needed by the other process. For example, transaction A obtains object 1, updates it, then attempts to obtain object 2. Transaction B first obtains object 2, and then attempts to obtain object 1. The transactions are deadlocked because they are waiting to obtain an object that is already locked, but neither transaction can release the object it currently has. As a result, the transactions are unable to continue processing. The transactions will eventually timeout because Oracle has deadlock detection.



Retrieve All Required Data for a Page at Once

If a non-configurable page has many objects and a lot of data to present to the user, you can optimize it by paginating the objects then getting the select information for all the objects on a single page. This technique allows you to retrieve the data for a small subset of the total number of objects. If the page supports column sorting, retrieve that one column at the same time as the OIDs. The entire data set for that column is required to perform a sort. A configurable page already works like this.

Applies to: JSP, Bean, JPO



Retrieve Data for All Table Rows in One Call (Program and ProgramHTML Columns)

For table columns configured to fire a JPO, pass the OIDs for each row so the JPO method can use DomainObject.getInfo() to get all the select information for all the objects in one call. This technique performs better than making a database call for each row in the table. Use this technique for relationship data as well.

Applies to: JPO



Transaction Boundaries

All pages that read from or write to the database should have transaction boundaries. This allows provides the benefit from the core caching mechanism/buffer rather than making a trip to the actual database for every piece of data.

Do not use the older transaction include files with the catch portion of try-catch block because this can lead to extra out.print("\n "); statements which (if they fail) can prevent the transaction from being aborted.

Add calls to transaction methods as follows:

```
try {
    ContextUtil.startTransaction(context, false); // Start
transaction at the start of the try block
    ...
    ContextUtil.commitTransaction(context); // Commit
transaction at the end of the try block
} catch (Exception ex) {
    ContextUtil.abortTransaction(context); // Abort transaction
```

```
inside of catch block  
}
```

Applies to: JSP, Bean, JPO



Transactions and Nesting

If you have multiple transactions, be careful about when you call the `start()` method on the context class. If you call the `start()` method of the context class before calling the `commit` or `abort` methods for the first transaction, the system throws a `MatrixException`. Before starting a transaction, use the `isTransactionActive()` method to determine if a transaction is still active. If the result is false, you can start another transaction. If true, you must abort or commit the active transaction before starting a new one.



Transactions and Integrations

To ensure that integration transactions fail gracefully when they encounter a locked resource, use the `set transaction nowait` statement in the `MQLCommand.executeCommand()` after `Context.start()`. The `nowait` statement generates an error if a requested object is in use. For example:

```
Context ctx=new Context(":bos", "localhost");  
:  
:  
ctx.start();  
MQLCommand cmd=new MQLCommand();  
cmd.executeCommand(ctx, "set transaction nowait");  
:  
:
```

This concept presents best practices for working with online help. The following topics are discussed:

- [Context-Sensitive Help](#)
- [Correct Approach](#)
- [Javadoc for JPO and Beans](#)

Context-Sensitive Help

Verify that context-sensitive help works and that help markers point to the correct page. See below for examples to call help from: Configurable Component, Frameset, and Custom pages.



Correct Approach

For configurable components, the system passes help markers as a URL parameter, for example:

```
emxTable.jsp?helpMarker=emxhelpcreatecollections
```

For Frameset pages, the system defines and passes help markers to the Frameset initialization method, for example:

```
String HelpMarker="emxhelpmecreate";
fs.initFrameset(heading, HelpMarker, contentURL, false, true,
false false);
```

For custom JSPs, make calls directly to the JavaScript methods, for example:

```
openHelp(helpMarker, directory, langStr);
openHelp("emxhelpcreatecollections", "engineeringcentral", "en");
```

Applies to: JSP



Javadoc for JPO and Beans

Add Javadoc for Bean methods and JPOs based on the guidelines.

Applies to: Bean, JPO

General

This concept presents general best practices.

The following topics are discussed:

- [Do Not Use Deprecated Elements](#)

Do Not Use Deprecated Elements

Do not use deprecated elements such as schema or methods. Appendix C in the *Schema Reference Guidelists* deprecated schema.

Applies to: JSP, Bean, JPO

HTML Tags

The following topics are discussed:

- [Close All HTML Tags](#)
- [Display/Visible Pages](#)

Close All HTML Tags

Close all HTML tags (for Netscape). Use common include files to ensure this. If you use the common include pages correctly, the begin and end tags (i.e. <body>, <html>, etc.) are automatically inserted. You can also use the common include files on processing pages.

The pages are as follows:

- emxUICommonHeaderBeginInclude.inc--Contains <html>, <head>, style sheet includes and common java script includes.
- emxUICommonHeaderEndInclude.inc--Contains </head>, <body> and the "Fields is red italics are required" message.
- emxUICommonEndOfPageInclude.inc--Contains </body> and </html> tags.

Applies to: JSP



Display/Visible Pages

For display/visible page, make sure all the necessary HTML tags are well-formed and in the correct locations.

Example:

```
<html>
<header> <title> xyz </title> </header>
<body> <form> ...
</form> </body> </html>
```

Applies to: JSP

This concept presents best practices for displaying data using i18n.

The following topics are discussed:

- [Admin Names](#)
- [Dates](#)
- [Text Display](#)

Admin Names

Use i18n for displaying Admin Names. For example the state names, attribute range values, type names, and so on, must be translated.

Applies to: JSP, Javascript



Dates

You must internationalize date displays using the <IzDate> taglib or the recommended methods in ematrixDateFormat Bean.

Applies to: JSP, Bean, JPO



Text Display

Use i18n methods for text display. Both Java and JavaScript code must use the i18n methods to handle any text string displayed to the user. All Javascript alerts have i18nScript tags.

Applies to: JSP, Javascript

The following sections describe the correct usage of the Izdate taglib methods.

The following topics are discussed:

- [Taglib Usage](#)
- [JavaScript Comparisons](#)

Taglib Usage

Use the following methods to convert the server date/time to client date/time into Java Date.

```
int iDateFormat =  
eMatrixDateFormat.getEMatrixDisplayDateFormat();  
  
java.text.DateFormat formatter =  
DateFormat.getDateInstance(iDateFormat, iDateFormat,  
request.getLocale());  
  
String tempDate =  
eMatrixDateFormat.getFormattedDisplayDateTime(context,  
sDueDate, true, intDateFormat,  
clientTZOffset,request.getLocale());  
  
Date dDueDate = formatter.parse(tempDate);
```

Do not use this technique:

```
Date dueDate = new Date(stringDate) (Not supposed to use this as  
this is deprecated)
```

This works only when stringDate is medium format or short format only in US format. Use getFormattedInputDateTime instead of using inline date manipulations or conversion to server time zone. However, this might not work for some non-US formats.

Use this technique:

```
eMatrixDateFormat.getFormattedInputDateTime(context,routeSchedu  
ledCompletionDate,routeTime[i],clientTZOffset,request.getLocale  
());
```

Do not use this technique:

```
strDateTime = routeScheduledCompletionDate + " " + routeTime[i];  
Date datel = new Date(routeScheduledCompletionDate);  
routeScheduledCompletionDate = (datel.getMonth()+1)+"/  
"+(datel.getDate())+"/"+(datel.getYear()+1900);  
routeScheduledCompletionDate = routeScheduledCompletionDate + "  
" + routeTime[i];  
//Formatting Date to Ematrix Date Format //strDateTime =  
eMatrixDateFormat.getFormattedInputDateTime(context,routeSchedu  
ledCompletionDate,routeTime[i],clientTZOffset,request.getLocale  
());  
strDateTime=routeScheduledCompletionDate;
```

Applies to: JSP, Bean, JPO



JavaScript Comparisons

Javascript does not support different formats for comparison. You can compare the date in Javascript by passing the date string in US format or in milliseconds.

Using US Format date string in Javascript:

```
java.text.SimpleDateFormat USformatter = new  
java.text.SimpleDateFormat ("MM/dd/yyyy hh:mm:ss a");  
  
String tempDate =  
eMatrixDateFormat.getFormattedDisplayDateTime(context,  
sDueDate, true, intDateFormat,  
clientTZOffset,request.getLocale());  
  
Date dDueDate = formatter.parse(tempDate);  
  
String maxDueDateForJS = USformatter.format(dDueDate);  
<script language="Javascript" > var maxDueDateForJSValidation =  
new Date(maxDueDate);</script>
```

Using Millisecond date in Javascript:

```
<script language="Javascript" > var DtMm = new Date();  
DtMm.setTime(eval("document.TaskSummary.routeScheduledCompleti  
onDate0_msvalue.value"));  
str4=document.TaskSummary.routeTime.value;
```

```
hrs4=parseInt(str4.substring(0,str4.indexOf(':')));  
mns4=parseInt(str4.substring(str4.indexOf(':')+1,str4.indexOf('')));  
if ( (str4.substring(str4.indexOf(' ')+1)=='PM') ) {  
hrs4=hrs4+12 ; } DtTm.setHours(hrs4);  
DtTm.setMinutes(mns4);  
</script>
```

Use Lzdate tag or proper date conversion methods in all pages that display the date field including: Property pages, Summary pages, Create/Edit Dialogs, and so on.

In UI forms and Tables--Lzdate tag is a property setting.

Non UI components--Requires some analysis for the correct date display technique.

Applies to: JSP, Bean, JPO

JavaScript

This concept presents best practices for writing JavaScript.

The following topics are discussed:

- [Inline Code](#)
- [Single Quotes in Messages](#)
- [Using Includes](#)

Inline Code

Avoid any inline JavaScript code within the JSP page. Move all the JavaScript code to a JavaScript file and then include the JS file in the JSP.

Applies : JSP, Javascript



Single Quotes in Messages

Do not use single quote in JavaScript messages or alerts and confirms. Use double quotes instead. This allows the use apostrophes in names.

Applies to: Javascript



Using Includes

Place all JavaScript Code includes and Style includes within the <header> or the <body> tag.

Example:

```
<header>
JS include/code...
</header>
<body>
JS include/code...
</body>
```

Applies to: JSP

This concept presents best practices for writing JSP pages.

The following topics are discussed:

- [Capturing Exceptions](#)
- [Forward Tags](#)
- [getPersons\(\) Method](#)
- [Restrictions](#)

Capturing Exceptions

Capture the exception in JSP and add to emxNavErrorObject. Make sure all the Java code is in a try-catch block. The catch block must use the emxNavErrorObject so that the page shows up an alert message on any error condition.

For example:

```
try {  
} catch (Exception ex) {  
    if (ex.toString() != null && ex.toString().length() > 0)  
    {emxNavErrorObject.addMessage(ex.toString());}  
} finally {  
    // Add cleanup statements if required like object close,  
    cleanup session, etc.  
}
```

Applies: JSP



Forward Tags

Remove response.sendRedirect() calls to JSP forward tags, or to a form submit. They fail in an SSO environment.

Applies to: JSP



getPersons() Method

Storing data in a HttpSession or static member variables of a JSP requires careful planning. For example, you might write a JSP to display a list of users in a select field using the `matrix.db.Person.getPersons()` method every time the page is displayed. If the installation has many users, a large number of objects that exist for a short period of time are created. This can strain Java garbage collection. A better solution is to retrieve the list once, keep it in a static member variable, and refresh it every so often.



Restrictions

Do not write Java code with business logic inside JSP pages. Put the business logic and major Java processing in Bean code. Call the bean methods from the JSP.

Applies to: JSP

Memory Leaks (IE)

This concept presents best practices to avoid memory leaks.

The following topics are discussed:

- [Cross Page References](#)
- [Circular References](#)
- [Closures](#)

Cross Page References

Leaks that are based on order of insertion are typically caused when you create intermediate objects that are not properly cleaned up. That is the case when creating dynamic elements and then attaching them to the DOM. In 1 pattern you attach 2 dynamically created objects together temporarily that creates a scope from the child to the parent element. Later, when you attach this 2-element tree to the primary tree, they both inherit the scope of the document and a temporary object is leaked.

Another pattern is when you attach elements into the primary tree working your way from top-level dynamically created element through all of the children. Because each attachment inherits the scope of the primary document, you never generate temporary scopes. This method is much better at avoiding potential memory leaks.

Applies to: JSP, Javascript



Circular References

When mutual references are counted between Internet Explorer's COM infrastructure and any scripting engine, objects can leak memory.

Circular references are the root of nearly every leak. Normally, script engines handle circular references through their garbage collectors, but certain unknowns can prevent their heuristics from working properly. The unknown in the case of IE is the status of any DOM elements that a portion of script has access to. The script engine objects hold a reference to the DOM element and wait for any outstanding references to be removed before cleaning up and releasing the DOM element pointer.

To break the leak pattern, use explicit null assignments. By assigning null before the document unloads you are telling the script engine there is no longer an association between the element and the object inside the engine. The system can properly clean up references and release the DOM element.

Incorrect approach:

The following code shows an example on this pattern looks like in HTML, which can cause a leak by using a global script engine variable and a DOM element.

```
<html>
<head><script language="JScript">
    var myGlobalObject;
    function SetupLeak()
    {
        // First set up the script scope to element reference
        myGlobalObject =
document.getElementById("LeakedDiv");

        // Next set up the element to script scope reference
        document.getElementById("LeakedDiv").expandoProperty
=
            myGlobalObject;
    }
    function BreakLeak()
    {
        document.getElementById("LeakedDiv").expandoProperty =
null;
    }
</script>
</head>
<body onload="SetupLeak()" onunload="BreakLeak()">
    <div id="LeakedDiv"></div>
</body>
</html>
```

Applies to: JSP, Javascript



Closures

Closures are often responsible for leaks because they create circular references. It is not immediately obvious that parent function parameters and local variables will be frozen in time, referenced, and held until the closure itself is released.

Normally, a function's local variables and the parameters used when calling a function only exist for the lifetime of the function itself. With closures, these variables and parameters continue to have an outstanding reference as long as the closure is alive, and since closures can live beyond the lifetime of their parent function so can any of the locals and parameters in that function. In the example, Parameter 1 would normally be released as soon as the function call was over. Because we have added a closure, a second reference is made, and that second reference will not be released until the closure is also released. If you attach the closure to an event, then you must detach it from that event. If you attach the closure to an expand, then you must null that expand.

Closures are also created per call, so calling this function twice will create two individual closures, each holding references to the parameters passed in each time. Because of this transparent nature it is really easy to leak closures.

At times it may be necessary to use a closure, but they should only be used when truly necessary.

Incorrect approach:

The following example provides the most basic of leaks using closures:

```
<html>
    <head>
        <script language="JScript">
            function AttachEvents(element)
            {
                // This structure causes element to ref
                ClickEventHandler
                element.attachEvent("onclick", ClickEventHandler);
                function ClickEventHandler()
                {
                    // This closure refs element
                }
            }
            function SetupLeak()
            {
                // The leak happens all at once
                AttachEvents(document.getElementById("LeakedDiv"));
            }
            function BreakLeak()
            {
            }
        </script>
    </head>\

    <body onload="SetupLeak()" onunload="BreakLeak()">
        <div id="LeakedDiv"></div>
    </body>
</html>
```

Applies to: JSP, Javascript

Miscellaneous

This concept presents a series of miscellaneous best practices.

The following topics are discussed:

- [Filter Option](#)
- [Java Objects: Use DomainObject](#)
- [Methods: Multiple Method Calls](#)
- [MQLCommand.executeCommand\(\)](#)
- [Netscape 7: Do Not Use document.location.reload\(\)](#)
- [Pagination](#)
- [Query: Specify Vault](#)
- [Relationship: Pattern Usage](#)
- [Release Mouse from Modal Dialog](#)
- [Schema Names: Do Not Hard-Code](#)
- [Use ArrayList/HashMap](#)
- [Use: emxGetParameter](#)
- [Use: getFrameContext\(\) NOT getContext\(\)](#)
- [Use: findFrame\(\)](#)
- [Use: listHidden Frame](#)
- [Use: open and close Methods](#)
- [Use: ReleaseMouse\(\) Method in emxUIModal.js](#)

Filter Option

If using the Filter option, ensure that the filtering occurs before pagination, otherwise some pages may have very few or no objects to display, but will still have multiple pages.

Applies to: JSP



Java Objects: Use DomainObject

If you are using a DomainObject and BusinessObject for the same Java object, change BusinessObject references to DomainObjects.

Applies to: JSP, Bean, JPO



Methods: Multiple Method Calls

When you need to use the same method more than once to get the same data, store the return from the first call in a local variable, and use the variable for subsequent references to that data. This reduces the overhead of another Bean method call.

Applies to: JSP, Bean, JPO



MQLCommand.executeCommand()

MQLCommand.executeCommand() returns a null pointer unless you do an `mqlcommand.open(_context)` and `mqlcommand.close(_context)` before and after.



Netscape 7: Do Not Use document.location.reload()

Do not use `document.location.reload()` calls (for NS-7) - use `document.location.href = document.location.href` instead.

Applies to: JSP, Javascript



Pagination

Database calls should return the first page of data only, otherwise you risk returning large amounts of unnecessary data that can affect performance. This becomes less of an issue as you move to configurable components because pagination is built-in.

Applies to: JSP



Query: Specify Vault

When specifying a query, always specify the vault or list of vaults unless "all" vaults is a requirement.

Applies to: JSP, Bean, JPO



Relationship: Pattern Usage

Review all Expand Selects and getRelatedObject() methods to ensure that only the appropriate relationship(s) are used in the expand statements. Never use an open ended expand that does not specify actual relationship names because of serious functional and performance implications.

This type of mistake is difficult to find, and is typically introduced when you look up an incorrect symbolic name and pass a null as the relationship pattern. Another common mistake is not using a type pattern when you should, or using one when you should not.

Applies to: JSP, Bean, JPO



Release Mouse from Modal Dialog

When opening a new window as a modal dialog using the method "showModalDialog()", make sure to call releaseMouse() upon closing to refresh the opening page.

Applies to: JSP, Javascript



Schema Names: Do Not Hard-Code

Do not hard-code schema names in any code. State names, range values, Type names, and so on, all should be looked up and then translated.

Applies to: JSP, Bean, JPO



Use ArrayList/Hashmap

Use ArrayList instead of Vector if synchronization is not required. Vectors are synchronized and therefore take more time for processing. Use Hashmap instead of Hashtable for the same reason.

Applies to: JSP, Bean, JPO



Use: emxGetParameter

Use the emxGetParameter wrapper method for reading request parameters. For getting the request parameter, use the following Request method that is defined in the emxNavigatorInclude.inc include file.

Example:

```
emxGetParameter(request, "objectId");
```

Applies to: JSP



Use: getFrameContext() NOT getContext()

Do not use getContext(). Use getFrameContext() instead.

Applies to: JSP, Bean, JPO



Use: findFrame()

Use findFrame() to locate named frames. To locate any named frame, use the standard JavaScript method findFrame() defined in the file scripts/emxUiConstants.js. Do NOT use parent.parent or opener.parent.parent to locate a named frame.

Applies to: JSP, JavaScript



Use: listHidden Frame

When using the configurable table component, use the listHidden frame within the table for processing the JSP within the hidden frame. Make sure the Toolbar command or column setting which needs the Target Location is pointing to the hidden frame. Assign it as Target Location=listHidden.



Use: open and close Methods

Interfaces for stateless objects in the Studio Customization Toolkit no longer require a call to open() or close(), thereby improving performance. Stateless objects include:

BusinessObject	BusinessType	Cue
Filter	Format	Group
Page	Person	Policy
Query	Relationship	RelationshipType
Report	Resource	Role
Store	Set	Table
Tip	ToolSet	User
Vault		

The context that is stored in the stateless object is NULL after a call to close(). Using any methods in a stateless class that don't take a context as a parameter require a call to open().



Use: ReleaseMouse() Method in emxUIModal.js

Use the releaseMouse() method (emxUIModal.js) to release the mouse after opening a modal dialog and refreshing the opener frame.

MQL Syntax

This concept presents best practices for MQL syntax.

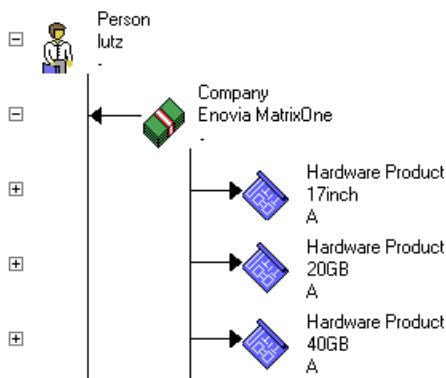
The following topics are discussed:

- [Selecting Admin Business Objects](#)
- [Modifying Objects](#)

Selecting Admin Business Objects

Performance can often be greatly improved by using the `object[]` selectable in place of complicated access expressions.

Associations between administrative objects are frequently modeled by a parallel set of business objects and connections. The most notable use of this is for user access to business objects. Typically, a business object of type Person is created for all Person administrative objects in the database; that is, all users. The Person business object is connected to business objects representing organizations to which the person belongs. These organizations are in turn connected to objects for which all individuals in the organization should have access rights, as shown below:



Which access rights are granted may be determined by access filters with complicated expressions that navigate these relationships. For example:

```
context.user ~~ to[Manufacturing  
Responsibility].from[from[Employee].to.name]
```

A more efficient way to perform this check is to compare a list of organizations the user belongs to with the names of the organizations to which the object is connected (That is, compare a list to a list, instead of finding an item within a nested list).

For this purpose, you can use the selectable, `object`, for all administrative objects. It can be used with the name of a business type in brackets, in which case it returns a business object of that type and with the same name as the administrative object. If no string is given in brackets, it finds the object of the business type that matches the administrative type with the case ignored (that is, the type could be person or Person or PERSON, etc.) In a case sensitive environment, if more than one type exists, such as Person and PERSON, the object with the latest originated date is returned. The expression shown above could be re-written using this selectable as follows:

```
context.user.object.to[Employee].from ~~ to[Manufacturing  
Responsibility].from
```

If there are multiple revisions with the same type and name, the latest one is returned, assuming they all belong to the same revision chain. If the objects of the same type and name are not part of the same revision chain, the one with the latest originated date is returned.



Modifying Objects

It is good coding practice to modify the order of table columns, form fields or any other workspace objects one at a time in separate MQL commands. Modifying multiple columns in a single MQL command could cause unexpected results to the end user.

Objects

This concept presents best practices for objects.

The following topics are discussed:

- [Administrative](#)
- [Query](#)
- [Session](#)

Administrative

Use symbolic names for referring to administrative objects. Do NOT hard-code any administrative object name in the JSP page (or any code for that matter). Use the symbolic name to lookup the actual name then use the value returned.

Example:

```
String personTypeName = PropertyUtil.getSchemaProperty(context,  
"type_Person");
```

Applies to: JSP, Bean, JPO



Query

If a page or Bean uses a Query object with a lot of Select statements and a Where clause, then change it to perform the query with just the Where clause. You can then just get the OIDs. This strategy requires an additional trip to the database, but performance is faster because the second call to the database should use DomainObject.getInfo(). This method call takes a list of objects and select statements and returns the results for all objects in one call. This is preferred because the query does the selects across all of the business objects found with the query before it applies the Where clause. This can result in very large amounts of data depending on how many objects the initial part of the query brings back.

Applies to: JSP, Bean, JPO



Session

Use HttpSession sparingly. Store multiple objects as a collection in the session instead of storing individual objects in the session. This reduces the number of method calls to store and retrieve values from the session. Clean up the session after use.

Applies to: JSP, Bean

Standard Pages

This concept presents best practices for working with standard pages.

The following topics are discussed:

- [Include emxNavigatorInclude.inc](#)
- [Include Standard Error Handling](#)

Include emxNavigatorInclude.inc

Include the standard emxNavigatorInclude.inc page as a static include. It contains all necessary Java imports, taglib, content type definitions, and logged-in check.

Example:

```
<%@include file = "emxNavigatorInclude.inc"%>
```

Applies to: JSP



Include Standard Error Handling

Include the standard error handling pages to the top and bottom of the page. The bottom include contains JavaScript code and hence must be included within <header> or <body> tags. For example:

```
<html>
<%@include file = "emxNavigatorTopErrorInclude.inc"%>
...
<%@include file = "emxNavigatorBottomErrorInclude.inc"%>
</header> OR </body>
</html>
```

Applies to: JSP

Statements

This concept presents best practices for issuing statements.

The following topics are discussed:

- [Block](#)
- [Debug](#)

Block

Use open and close braces for ALL block statements. All "if", "for" (even one line) statements need to be inside the braces. This is required both for coding standard purposes as well as for Tomcat.

Applies to: JSP, Bean, JPO, Javascript



Debug

Remove all debug statements. Remove all System.out() statements and all JavaScript debug alert statements before you submit any code. Do not simply comment them out; remove them.

Applies to: JSP, Bean, JPO, Javascript

StringBuffer

This concept presents best practices for working with the string buffer.

The following topics are discussed:

- [Concatenation](#)
- [Initialization](#)
- [Usage](#)

Concatenation

Never use StringBuffer concatenation with the plus sign (+) because of the severe performance hit for doing so. NEVER use strBuff.append("abc" + variable + "def"). Instead use three separate append operations.

Applies to: JSP, Bean, JPO



Initialization

Always initialize StringBuffer objects with a length sufficient to hold the entire string. For example:

```
StringBuffer strBuff = new StringBuffer(64);
```

Applies to: JSP, Bean, JPO



Usage

Use StringBuffer for string concatenation instead of String. This addresses the relative performance of the two techniques.

Applies to: JSP, Bean, JPO

Stylesheets

This concepts presents best practices for working with style sheets.

The following topics are discussed:

- [Do Not Use Inline Styles](#)
- [Use addStyleSheet\(\)](#)

Do Not Use Inline Styles

Do not use inline styles--use the common stylesheets. Do not use new stylesheets unless the Design Engineering group approves them.

Applies to: JSP



Use addStyleSheet()

Use addStyleSheet() method to include style sheets. If including any Style sheet, make sure the Style include uses the addStyleSheet() method.

Applies to: JSP

URLs

This concept presents best practices for writing URL strings.

The following topics are discussed:

- [Parameter Names](#)

Parameter Names

Avoid passing parameter names having spaces in the URL because they fail in SunOne/Netscape.

Applies to: JSP

Variables

This concept presents best practices for using variables.

The following topics are discussed:

- [Static Variables](#)

Static Variables

NEVER use a static Context variable in a JSP or Bean or JPO. It crashes the ENOVIA Live Collaboration Server when multiple threads use this variable.

Static variables do not get dereferenced during the entire life of the application, and therefore do not get garbage collected by Java. Use static variables ONLY if some data needs to be cached for the life of the application.

Applies to: JSP, Bean, JPO

WebSphere

This concept presents best practices for working with WebSphere.

The following topics are discussed:

- [Empty Strings](#)

Empty Strings

Initialize any string value that can be null (empty) as the empty string (""). If you do not, it gets displayed as null in WebSphere.

Applies to: JSP, Bean, JPO

Dynamic UI Parameters

This section contains alphabetic lists of parameters and settings used by dynamic user interface components. For details on configuring components and how to specify the parameters and settings for a particular component, see the section for that component.

In this section:

- [Administrative Object Parameters](#)
- [Settings](#)
- [URL Parameters](#)
- [Parameters Automatically Passed to URLs](#)

Administrative Object Parameters

This section lists parameters for the administrative objects that represent dynamic user interface components, such as menu and command objects.

You can define these parameters using the create and edit dialog boxes for the administrative objects in Business Modeler or using commands in MQL. The parameter names reflect the names used in Business Modeler and in some cases are different than the command used in MQL for the parameter. For information on the appropriate MQL commands to use, see the *MQL Guide*.

Parameter	Description	Accepted Values/Examples
UI Component		
Access Business Modeler tab / user MQL command, and parameter	The persons, roles, and groups who can access the UI component or link or filter control. When you assign a role or group, all child roles/groups also receive access. To make the component available to all users, regardless of role/group assignments, choose All. Note that if no users are assigned access, the system assumes all users have access. If a user does not have access to any links in a menu, the menu is not displayed. command objects for submenu links, toolbar tools, toolbar items, PowerView tabs table object columns toolbar link command object, tree category command objects	Names of group, role, person administrative objects. or All (default)
Alt command objects for toolbar tools, tree menu objects, table column objects, PowerView tabs	The text to display in the ToolTip that appears when a user moves the mouse pointer over the UI component. Either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it does not find one, it displays the entered text. Applicable for table columns only when the column type is set to Icon. The Alt text for a command is ignored. The Label, not the Alt value, displays when the mouse hovers over the tab. If the tab's label is truncated because it exceeds the limit, the full Tab label text is shown in the tooltip. When used with a custom toolbar filter, only use Alt if the Input Type = submit.	emxFramework.Home.Logout Change Password Name of what the icon represents.
Alt columns in a structure browser	Used for structure browser columns when the Column Type is set to programHTMLOutput or image. The text defined for this parameter shows in the mouse-over popup DIV. If this parameter is not defined, the popup DIV shows the contents of the cell.	Name of what the image or column data represents.
Applies To table object columns form objects	The item to which to apply the select expression. Applying an expression to a relationship gets a table of objects related to a known object. The system passes a valid business object ID or relationship ID to the JSP. For example, emxForm.jsp?objectId=xxx or emxForm.jsp?relId=yyy. Dynamic UI table and tree components pass the objectId and relId automatically to emxForm.jsp, so the href link	Business Object Relationship

	can be configured without passing IDs.	
Arguments	Used to replace macros within the code. The symbolic names will be translated into the actual name during substitution.	PART=type_Part EBOM_REL=relationship_EBOM ID = dummy
inquiry objects	ID is a reserved keyword for the inquiry objects used with the configurable table components. If the macro \${ID} is used in the code section, the argument ID must be assigned to any dummy value. This value is replaced by the request parameter "objectId" at run time.	
Channel	The channels associated with the portal object.	Names of channel objects, such as: AEFCollections AEFPowerViewChannel SPCSOChannel
portal objects		
Code		
inquiry objects	The code section determines the output object list for a specific business requirement. The code is generally an MQL temp query or expand bus command that selects the found objects' ids.	temp query bus \${PART} * * select id dump expand bus \${ID} from relationship \${EBOM_REL} select businessobject id select relationship id dump
toolbar link command objects	<p>The macro \${ID} is substituted with the business object id during run time. This object Id is available to emxTable.jsp, passed in from the tree node or any other commands that operate based on a specific business object. For example, typically when emxTable.jsp is called from a tree, the "objectId" parameter is passed with the specific OID assigned to the current tree node and that OID will get substituted with \${ID} at run time.</p> <p>Note that this \${ID} is not same as the RPE variable \${OID}.</p> <p>All the administrative types like Type names (Part, Buyer Desk), attribute names, relationship names must be defined as macros and each macro will be assigned with its symbolic name in the arguments list.</p> <p>All other macros (like \${PART}, \${EBOM_REL}) defined within the code are to be assigned to the appropriate symbolic names in the Argument, if required. The macros are evaluated at run time by converting the symbolic name to actual name.</p> <p>For toolbar command link objects, this is custom JavaScript code entered in the Code tab of the command. JSON object syntax must be used for this code.</p>	
Command (specified in the Items tab in Business Modeler)	The command objects that represent the links in the toolbar for the top-level menu and that represent links in the drop-down menu for lower-level menus. The order of the commands is the order the links appear on the toolbar and in the drop-down menu.	Names of command objects, such as: ENCCreateRevisionToolbarActionLink TMCEditProfileToolbarActionLink ENCBOMTreeCategory
toolbar menu objects	For trees, the command objects that represent the categories that should be in the tree.	AEFRelToItemTreeCategory SPCSOInstructions SPCRoutesTreeCategory ENCEffectivityDateFilter
tree menu objects	For channels, the command objects that represent the tabs in the channel. The order of the commands is the order the tabs appear in the channels in the user interface.	
channel objects		
Description	Use to give a brief description of how to use the form or field. The web form administrative object and each field within it has a description parameter.	Text string
form objects, toolbar textboxes		
Expression	The select expression to get the column or field data. This expression is applied to either the relationship or business object, as specified in the Applies To options.	For business objects: type name \$<attribute[attribute_Originator].value>
table object columns, form object, generic create form object	Note that the Expression and Applies to options in Business are equivalent to the businessobject and relationship MQL commands.	For relationships: \$<attribute[attribute_FindNumber].value>

		\$<attribute[attribute_Qty].value>
Format inquiry objects	The format determines how that line is to be reformatted before printing the output.	If the table column is defined based on a list of business object IDs, the format must be a list of business object IDs separated by a new line. The format value should be \${OID} and an example output is: 12333.3453.56765.3443 12533.3453.56765.3453 12633.3453.56765.3943 ? If the table column is defined based on business object and relationship IDs, the format must be a list of business object and relationship ID pairs, separated by the new line character. In each pair, the relationship ID comes first and is separated by the business object ID by a ~. The format value should be \${RELID}~\${OID} and an example output is: 9089.34345.56567.21312~12333.3453.56765.3443 4564.3445.567.7868~12533.3453.56765.3453 6465.2342.4566.3212~12633.3453.56765.3943 ?
Header History page	The text to use for the header of the History page. This parameter can either be a string resource ID or a mixture of macros and text or simply text.	\$<type> \$<name> \$<revision> emxFrameworkStringResource.Common.HistoryPageHeading Object History
Heading (label MQL command)	The text that should appear on the column header. Either a string resource ID for the text string or the actual text string that should appear.	emxEngineeringCentral.common.Name emxTeam.Common.ProjectName Description
table column objects	The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it does not find one, it displays the entered text.	
Height channel objects	The height of the channel in pixels.	The default is 260.
hidden table column object	Hides or shows a column. If a column is hidden (true), it does not show up in the default system table for an object type.	true false (default)
href toolbar command objects My Desk and Actions menus menu objects for navigation trees table object columns form objects subscription commands	The URL to call when a user clicks the data, label or image that represents the UI component, or submits input for a custom toolbar filter. An href is not applicable to menu objects for My Desk, Actions, or Toolbar menus or submenus. The Toolbar menu cannot be clicked (only the individual tools can be clicked). The submenu labels for My Desk and Actions menus can be clicked but doing so only expands and contracts the menu and does not call another page. For menu objects that represent the root node for a tree, the URL that gets executed when a user clicks the root node for the tree, which is the Properties page for the object by default. This page is always displayed in the right frame, which is called "detailsDisplay" and not in another window or frame. The possible values/cases are: A JSP and any associated parameters. You can specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. A JavaScript function. A JSP with embedded macros and expressions. For forms, this parameter is used only for the View mode. In Edit mode, it is ignored because fields in Edit mode are not shown with a hyperlink. For forms, when a user clicks the hyperlinked field data, the system passes the objectId parameter as part of the URL. By default, the value for the objectId parameter is the ID of the business object the form page applies to. Using the Alternate OID expression setting, you can	emxLogout.jsp \${SUITE_DIR}/emxpartCreatePartDialog.jsp?mode=Promote&id=0 javascript:window.open('emxChangePassword.jsp', 'ChgPwd', 'width=400,height=400') javascript:showModalDialog('emxCreatePart.jsp', '400', '400') javascript:showModalDialog('\${SUITE_DIR}/emxCreateSketch.jsp', '600', '600') \${COMMON_DIR}/emxPartDetails.jsp \${COMMON_DIR}/emxTree.jsp javascript:window.open('emxEditPart.jsp', 'EditPart', 'width=400,height=400') javascript:top.historyControl.goBack() \${COMMON_DIR}/emxTree.jsp \${SUITE_DIR}/emxpartEditPartDialog.jsp

	configure the field to pass the ID returned from a different business object.	
href	To be assigned to the search JSP page to be used for displaying the search content. This specific JSP file must contain a JavaScript method called "doSearch()", to initiate the search process by calling a processing page. All original URL parameters passed-in when opening the search window are preserved and passed down to the specific search pages. The processing page must refresh the whole search window with the results page. Results page may use the AEFSearchResultsToolbar as the toolbar for this page to leverage "Revise Search" and "Add to Collection" support.	\${SUITE_DIR}/emxTeamFilesSearch.jsp??.
href	The URL to display when a user selects the tab. Tabs can display content from the following sources: configurable table configurable form JSP page URL If the specified URL contains a header parameter, it is not used because the header would be redundant with the tab label. For example, if the URL contains emxTable.jsp with a header specified, the table header is not used.	emxTable.jsp?xxxxx emxForm.jsp?xxxxx Sample.jsp?xxxxx http://www.matrixone.com
Icon	Icon for the object within ENOVIA Live Collaboration. The dynamic user interface does not display images for each application.	The name of an image file, such as Part.gif.
tree menu objects	This setting is NOT the image that displays next to the root node on the dynamic user interface. The emxFramework.smallIcon.SYMBOLIC_NAME property in emxSystem.properties is used to specify that image.	
Label	Form: Use to enter the text that should appear as the label for the field. Make sure Custom Label is checked so the system gets the label you specify. If Custom Label is unchecked, the system uses the expression as the label. Toolbar Textbox: Defines the text to display on the toolbar link. You can configure a link to have a text label, an image, or both. If you do not specify an image using the Image setting, you must specify a label. When used to define a textbox command on the toolbar, the Action Label and Input Type settings must also be defined. Either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text.	emxEngineeringCentral.common.Name emxTeam.Common.ProjectName Description
Label	Not applicable for top-level menus. The text that should be used to represent the UI component on the interface. For example, the text to represent the submenu, link in a menu, tree or tree category, or toolbar item. Either a string resource ID for the text string or the actual text string that should appear. To internationalize the text, you must use a string resource ID. The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text. For a custom toolbar filter, defines the text that precedes the Input Type control.	\$<type> \$<name> \$<attribute[attribute_Weight].value> Engine - \$<type> \$<name> Connected ECR emxFramework.Common.Part \${TYPE} \${NAME} If the label text is not found and no image is defined, the toolbar component displays an error icon to indicate that a configuration error has occurred.

Label	<p>The text that should be used to label the tree root node or category. The label can contain be defined using:</p> <p>Any of the regular expression macros supported by applications are evaluated at run time to display the current object type name and revision.</p> <p>Plain text to be displayed.</p> <p>String resource ID defined in the string resource property file. The system initially considers every label text as a string resource ID. If it finds the value for this ID in the resource file, it displays the obtained value. If not, the text is displayed as is.</p> <p>There are two other ways to define the label for a tree node: using the TreeLabel URL parameter passed to emxTree.jsp and using a JPO (specified with the Label Program and Label Function settings). The order of precedence in which the tree handles these methods of getting the root node label is:</p> <ul style="list-style-type: none"> TreeLabel URL parameter--Overrides all other methods. JPO--Used if TreeLabel parameter is not defined. Label parameter--Used only if TreeLabel and JPO are not defined. <p>The labels for tree categories can include a count of the number of items in the category.</p> <p>Tree category labels can also be defined using a JPO. See the Label Function and Label Program settings. If these JPO settings are defined for a category, they override any label defined using this parameter. The TreeLabel URL parameter for emxTree.jsp also overrides any label specified for the root node menu object.</p>	<pre>\$<type> \$<name> \$<attribute[attribute_Weight].value> Engine - \$<type> \$<name> Connected ECR emxFramework.Common.Part</pre>
Label	<p>The text that should display on a PowerViewtab. The label can be made up of:</p> <p>Plain text, such as "ECRs".</p> <p>A string resource ID, such as "emxEngineeringCentral.common.portal".</p> <p>To internationalize the text, you must use a string resource ID. The system first looks for a string resource ID that matches the entered value. If it finds one, it uses the value for the ID. If it doesn't find one, it displays the entered text and/or output from the select expression.</p> <p>The label can be up to 17 characters in length. If it exceeds 17 characters, it is truncated to17 characters and an ellipsis ("?") is added to the label. This default can be changed in emxsystem.properties with the property emxFramework.PowerView.Channel.Label.MaximumLength. The HTML "Title" attribute is defined so that when the user mouses over the tab, the full label displays.</p>	<pre>\$<type> \$<name> \$<attribute[attribute_Weight].value> Engine - \$<type> \$<name> Connected ECR emxFramework.Common.Part \${TYPE} \${NAME}Properties EBOM Lifecycle emxEngineeringCentral.common.portal</pre>
Menus (specified in the Items tab in)	<p>The menu objects that represent drop-down menus. The order of the menus is the order they appear in the toolbar or drop-down menu.</p> <p>Not applicable to menu objects for trees because trees do not have submenus.</p>	Names of menu objects, such as: ECBOMListToolBar TMCProjectsToolBar ENCFilterMenu
toolbar objects	When defining custom filters, the list of menus to add to the toolbar. Menu names can use the underscore (_) or hyphen (-) characters, but no other special characters.	
Name	Name of the table, table column, inquiry, channel, or portal administrative object. For inquiry objects, this is the name that will be referenced to evaluate this inquiry within a JSP.	Name Revision Originator ENCBOMList ENCCustomFilter
table objects, table object columns, inquiry objects, form objects, portal objects,channel objects, toolbar	Use to enter the name of the field used as identifier for the field within the web form object. Use to enter the name of the web form administrative object. When using multiple textbox commands in the toolbar, the	

textboxes, subscription commands	<p>Name distinguishes the specific textbox.</p> <p>When used as a filter, the name of the filter toolbar item. The JPO and any custom JSP use this parameter to get the value entered by the user. The Name can include the underscore (_) or hyphen (-), but no other special characters.</p> <p>When naming a column that will be used by a structure browser for calculations, you cannot include a space character in the Name.</p>	
Pattern inquiry objects	<p>Indicates the expected pattern of the results of the evaluated code and shows how the output should be parsed. It sets the desired field to an Runtime Program Environment (RPE) variable or macro. The pattern is applied to each line of output from the code.</p>	<p>* * * \${OID} for a list of business object IDs</p> <p>* * * * * \${OID} \${RELID} for a list of business object and relationship ID pairs</p>
RangeHref (range MQL command)	<p>For a table column or form field in , the RangeHREF parameter is set on the Link tab.</p> <p>Use to configure a textbox that has a Browse (...) button that calls a chooser or custom window from which users can select a value to populate the textbox.</p> <p>In the RangeHref parameter, specify the href URL to display the window. For example, the href might call a custom selection page.</p> <p>Range helpers are used for table or form pages only in Edit mode. The only control that can be configured with a range helper is textbox. To specify the control type, set Input Type=textbox.</p>	<p>Specify the path of the JSP using any of the standard directory macros or you can leave off the path designation to use the registered directory. For example:</p> <p> \${COMMON_DIR}/emxSelectVault.jsp</p> <p> \${SUITE_DIR}/emxSelectUser.jsp</p> <p> emxTypeChooser.jsp?typeList=type_Part,type_Document</p> <p> ../common/emxTypeChooser.jsp?</p> <p> &SelectType=multiselect&SelectAbstractTypes=true&InclusionList=eServiceEngineeringCentral.Types&observeHidden=true&>ShowIcons=true</p>
renderPDF form objects	<p>Parameter to control whether or not to show the Render PDF icon in the form toolbar when displaying the form in view mode.</p> <p>If the parameter is not passed in, this parameter is assumed false and the icon is not shown.</p>	<p>false (default)</p> <p>true</p>
Settings toolbar objects, tree menu objects, table column objects, form objects, command objects for PowerView tabs	Additional settings that define the behavior and appearance of the toolbar menu objects.	Name/value pairs, as defined in Settings .
sorttype	Defines how the column values should be sorted.	numeric--Sorts the column values numerically. alpha (Default)--Sorts the column values alphabetically. other--Sorts the column values using a custom algorithm defined as a JPO and specified in the Sort Program setting for the column.
table object columns	<p>This parameter currently is not available in Business Modeler and can only be set using the sorttype command in MQL. For example:</p> <pre>add table TestTable1 system column name Type label emxEngineeringCentral.Part.Type user all businessobject type setting "Registered Suite" EngineeringCentral sorttype numeric ;</pre>	
Test inquiry objects	<p>Use the Test tab to determine if the inquiry will parse the output as you have designed the JSP to expect to receive it. To test the expression entered on the Code tab, click Evaluate. The output displays on the tab. If runtime arguments are needed, enter them in the Input box. If no input is provided, the system uses the arguments entered</p>	--

on the Arguments tab.

You cannot use the Test tab if there are symbolic names in the code. For testing purposes, you can substitute the symbolic names in the code with actual names and then click on Evaluate.

If the inquiry is an expand query, \${ID} must be substituted with the actual objectId (for testing only).

Settings

This table lists and describes the settings for administrative objects that represent UI components. Note that the name and value for each setting are case sensitive.

Setting UI Component	Description	Accepted Values/Examples
Access Expression command objects for menus and toolbar items, PowerView tabs, table columns, form fields, generic create forms, configurable search form, tree categories, structure browser pages	<p>Controls access to the component based on a valid MQL expression. The expression gets evaluated at runtime. If the expression evaluates to True and no other access control prevents access, the component is shown, otherwise it is hidden.</p> <p>Depending on the type of expression defined, the program may or may not need a valid objectID.</p> <p>Administrators need to make sure that an objectID is available before configuring this setting for configurable toolbar menus and commands.</p> <p>When the setting is used for commands connected to the AEFGlobalToolbar (My Desk, Actions, Tools), the expression is evaluated on the person business object. It does not depend on any objectID to be passed in as a URL parameter.</p> <p>In cases where no objectID is passed, such as a table page originating from the My Desk menu, the Access Mask setting is ignored. Similarly, the IconMail tree, which does not have an objectID, does not use this setting if defined.</p> <p>Generic Create Form: Do not use object-specific expressions.</p> <p>Access Expression is not supported for non-object based structure browsers.</p>	<pre>attribute[\$<attribute_Weight>].value > 100 The toolbar item displays only if the Weight attribute on the business object is greater than 100. attribute[\$<attribute_Originator>].value == owner For menu command objects: context.user.name=="Test Everything" context.user.isassigned[Employee] == TRUE</pre>
Access Function command objects for menus and toolbar items, PowerView tabs, table columns, form fields, generic create form, configurable search form, tree categories, structure browser pages	<p>The name of the JPO method to invoke in the JPO specified for the Access Program setting. The Access function gets the input parameter as a HashMap, which contain all the request parameters that were passed into the Form page. The JPO method must return an object of class Boolean. If the returned value is true and no other access control prevents access, the component is displayed. If false, it is hidden.</p> <p>This is particularly useful if there is a criteria for access has nothing to do with users' roles. For example, suppose a link should only be shown to users who are employees of the host company. The JPO and method might check the user's company and display the link only if the user is from the host company.</p> <p>Generic Create Form: Do not use object-specific expressions.</p>	<p>The name of an access check method in the JPO specified in the Access Program setting, such as:</p> <pre>emxAccessCheck()</pre>
Access Mask command objects for menus and toolbar items, PowerView tabs, table columns, form fields, tree categories, structure browser pages, subscription commands	<p>Specifies the accesses the user must have for the current business object in order for the component to be displayed.</p> <p>When the setting is used for commands connected to the AEFGlobalToolbar (My Desk, Actions, Tools), the expression is evaluated on the Person business object. It does not depend on any objectID to be passed in as a URL parameter.</p> <p>When the setting is used for commands in page toolbars, menus, trees, table columns, and form fields, the access mask is evaluated on a specific business object. This business object is available only when the objectID is passed in as a URL parameter to the JSP.</p> <p>Administrators need to make sure that an objectID is</p>	<p>Any set of accesses, separated by a comma. For example:</p> <ul style="list-style-type: none"> Modify Delete ToConnect ToDisconnect FromConnect FromDisconnect Modify,Delete FromConnect,FromDisconnect

	<p>available before configuring this setting for configurable toolbar menus and commands.</p> <p>In cases where no objectId is passed, such as a table page originating from the My Desk menu, the Access Mask setting is ignored. The IconMail tree, which does not have an objectId, does not use this setting if defined.</p> <p>If the user does not have all the specified accesses in the business object's policy for the current state, the component is hidden. If the user has the access and no other access control prevents access, the component is displayed.</p> <p>You can specify multiple accesses by separating the accesses with a comma.</p> <p>Access Mask is not supported for non-object based structure browsers and toolbar custom filters.</p>	
Access Program command objects for menus and toolbar items, PowerView tabs, table columns, form fields, generic create form, configurable search form, tree categories, structure browser pages	<p>Controls access to the UI component based on the output from a method in the specified JPO program. The program must be defined in ENOVIA Live Collaboration. This setting requires that the Access Function setting also be specified. If Access Function is not set, the Access Program setting is ignored.</p> <p>The following input values are required for the program:</p> <ul style="list-style-type: none"> A list of all of the request parameters in a HashMap Method name as a string JPO Program Name as a string Context The output must be a Boolean. <p>Generic Create Form: Do not use object-specific expressions.</p>	<p>Name of a JPO defined as a program object in ENOVIA Live Collaboration, such as: emxAEFCollectionAccess</p>
Action Label	Only used with toolbar commands if Input Type is combobox or textbox. If used, a submit button shows after the combobox or textbox and uses the string specified by this setting as its label. The button refers to the JSON code:href specified on the textbox or combobox, so that a second command is not required to configure the button.	emxFramework.SampleFeature.Find Find emxFramework.Common.Filter Apply Filter
Action Type command objects for toolbar items	Tells the configurable toolbar that the command is a separator instead of a link. If the setting is not included, the toolbar treats the command as a standard link.	Separator--A line that separates one or more links. If the command is assigned to the top-level menu and is therefore on the toolbar itself, it is considered a vertical separator. Otherwise, it is considered a horizontal separator.
Additional Query form object field, table column	When Type Ahead is configured on the field/column (by setting the RangeHref to emxFullSearch.jsp or using a predefined Type Ahead Chooser), this setting defines additional field/selection critiera used to restrict the selection list. See About Automatic Type Ahead .	<FieldName1>=<select expression1>:<FieldName2>=<select expression2>:
Admin Type table object column form object field generic create form fields configurable search form structure browser pages	<p>Use to translate columns or fields whose values are administrative types or ranges of attributes.</p> <p>Based on the administrative type, the value is translated and presented. For attributes, it is necessary to provide the symbolic name of the attribute, which starts with "attribute_" and is followed by the attribute name with no spaces.</p> <p>For example, suppose you are configuring a field that shows an object's current state and you want the state name to be translated. You would add this setting and set the value to State. The translations for administrative object names are stored in the emxFrameworkStringResource.properties files, as described in the Internationalizing Dynamic UI Components.</p>	<p>Type State Role Relationship attribute_UnitOfMeasure</p> <p>These keywords get the field values translated for the appropriate type name:</p> <p>Type State Role Relationship Policy</p>

		<p>Group</p> <p>Vault</p> <p>Attribute (for translating the attribute name, not the range values)</p> <p>To translate an attribute value and range values, specify the symbolic name of the attribute. For example:</p> <p>attribute_UnitOfMeasure</p> <p>attribute_PartClassification</p>
Allow Manual Edit	When true, users can manually edit the form row or table column. Applicable only when the range parameter is set to a URL or when the setting format is assigned to date or for fields of type combobox. It is ignored in all other cases.	false (default)--Manual entry is not allowed. true--Manual entry is allowed.
form object field generic create object form field table object column configurable search form structure browser pages toolbar link commands	When this setting is true, the Admin Type setting is ignored. For a toolbar link, only used when the format=chooser setting is defined, which is only used when the Input Control = textbox setting is defined.	
Alternate OID expression	By default, when a column's data is configured to show as a hyperlink using the href parameter, the system passes the ID of the object for that row in the objectId parameter (tables) or the business object the form page applies to. Using this setting, you can configure the hyperlink so a different objectId is passed. The system passes the ID for the object returned from the expression defined in this setting.	\$<to[relationship_NewPartPartRevision].from.id> \$<to[relationship_EBOM].from.id>
table object column structure browser pages, form object field		
Alternate Policy expression	This setting is required to display the state of any connected objects and show the value translated.	Any select expression that evaluates to a policy of a connected object.
table object column, structure browser pages	This setting is applicable only when the Admin Type setting is set to State.	
Alternate Type expression	When the Show Alternate Icon setting is true, this expression is used to obtain the object type. Based on the obtained type, the corresponding icon is displayed.	\$<to[relationship_NewPartPartRevision].from.type> \$<to[relationship_EBOM].from.type>
table object column, form object field, structure browser pages		
Auto Filter	Determines whether users can filter the table rows based on data in the column. If at least one column in the table has Auto Filter set to true, the Filter  tool displays in the page toolbar. This setting cannot be applied to any column that has multiple values for a single data column cell. Configuring many columns for auto filtering or applying auto filter to inappropriate columns can potentially have high performance impact when viewing the filtered list. For optimum performance, define a limited number of columns with Auto Filter. Also, choose columns that have a limited set of possible values on which the user may want to filter the table data. For example, state and owner columns. Columns such as description and name are inappropriate because of the unlimited number of values that will result. If passed, the autoFilter URL Parameter overrides this setting. Filtering applies to root level objects only.	true--The Filter tool displays in the page toolbar. When clicked, the column is available on the Auto Filter Selection page. false (default)--The column is not available in the Auto Filter Selection page.
Access Behavior	This setting works with the results of the Access	hide (default)

	<p>Expression, Access Mask, or Access Program/Access Function settings. Those settings return a true or false value.</p> <p>When true, the menu command shows in the list and can be selected by the user. When false, this setting defines whether a menu command is hidden from the user (hide; does not display in the menu) or shown in gray (disabled) so that it cannot be selected.</p> <p>This setting does not work with Role accesses. Commands not available to a user based on role are always hidden, regardless of the value of this setting.</p>	disabled
Toolbars		
Calculate Average table column objects	A setting of true shows the Average of the column values. This is calculated using the total of the values in the column divided by number of rows. Default Label is Average.	true false
Calculate Sum table column objects	A setting of true shows the total of the column values. Default Label is Total.	true false
Calculate Maximum table column objects	A setting of true shows the largest number of the column values. Default Label is Maximum.	true false
Calculate Minimum table column objects	A setting of true shows the smallest number of the column values. Default Label is Minimum.	true false
Calculate Median table column objects	<p>A setting of true shows the middle number of the column values. If there is an even number of values, the average of the two middle values is shown.</p> <p>The median value for the odd number of rows is not rounded since no calculation is done. Default Label is Median.</p>	true false
Calculate Standard Deviation table column objects	<p>A setting of true shows the standard deviation of the column values. The formula used for standard deviation is:</p> <pre>sd = square root [sum(x - xbar)^2/ (N-1)]</pre> <p>Where</p> <p>N is the total number of elements.</p> <p>xbar is the mean of the column values</p> <p>Default Label is Standard Deviation.</p>	true false
Calendar Function table column objects, structure browser column objects, form field objects	Use to specify the name of the method in the JPO specified in the Calendar Program setting that retrieves the non-working days based on the calendar defined for the location.	The name of a function in the Calendar Program JPO, such as: <code>getNonWorkingDays</code>
Calendar Program table column objects, structure browser column objects, form field objects	Use to specify the name of a JPO that contains a method to get the non-working days for a calendar.	The name of a JPO, such as: <code>emxWorkCalendar</code>
Cols form object field configurable search form	Used when the input type is set to textarea. This setting limits the length of the textarea on the form and specifies the visible width in average character widths. If not specified, it uses the HTML default, which is 25.	25 40 50
Column Count create form fields	The number of name/value columns to draw horizontally. For Create forms, this setting applies only to a field defined with Field Type = Dynamic Attributes.	1 (default) 2 3

		<i>n</i>
Column Icon table object column, structure browser pages	Use to display an icon for the column's data instead of other data. Required when the setting "Column Type" is assigned to "icon".	Name of an image file such as: images/NewWindow.gif images/EditItem.gif
Column Type table object column structure browser pages	The setting "Column Type" is used when no expression is defined for the column data.	<p>program--The values for this column are obtained from a program (JPO). With this setting, the program and function name are required as settings.</p> <p>programHTMLOutput--Same as the "program" setting above, except that the column value output is in HTML format. Column values are placed in table cell between <td> and </td> tags. This setting ignores other column settings such as Show Type Icon, href, format, and Alternate OID expression.</p> <p>icon--Used when the column values are shown as an icon. The setting "Column Icon" must be defined with the icon to be displayed.</p> <p>image--Used when the column values are images. When used, the column shows the primary image associated with the business object (cannot be used with Structure Browser).</p> <p>checkbox--Used when the column values are check boxes shown grayed or not-grayed based on the access to the object in that row. This access can be based on business logic and defined in a JPO or it can be role-based and defined by assigning roles to the column. If the checkboxes have no access restrictions, this setting is not required and you can create checkboxes by passing in the parameter selection=multiple to emxTable.jsp.</p> <p>separator--Used to define a column of white space between standard data columns. A separator is especially useful to separate two groups of columns.</p> <p>file--Shows  which hyperlinks to a Quick File Access page listing files checked into or connected to the object. The optional Relationship Filter setting defines how to locate related files. If Common Components is not installed, this value is ignored.</p>
Comparable column in table	Used only for a table showing a structure compare report. Defines whether the column can be used as a comparison criteria in a structure compare report. By default, all columns are comparable except for Column Types of: Image Icon Separator	true (default) false
Compare Report column in table	Used only for a table showing a structure compare report. Defines whether to show the column in the structure compare report. By default, all columns are shown except for Column Types of: Image Icon Separator	hide show (default)
Confirm Message command object for toolbar items	Provides a JavaScript confirmation message when users click on the toolbar item. For example, a custom delete message can be configured for the onClick event to display a JavaScript confirm message to the user. The Confirmation dialog has the "OK" and "Cancel" button. Clicking OK proceeds with the processing and Cancel	<p>The actual text to display in the confirmation message or a string resource property key. To internationalize the message, a string resource key must be used:</p> <p>Are you sure you want to delete this object?</p>

	cancels the operation.	emxFramework.common.alertMsg
Create Exclude form object fields	Comma-separated list of attributes that will not be included in the webform when opened in Create mode when a field on the webform has been defined as Field Type = Dynamic Attributes.	For example: attribute_Cost
Currency Converter menu objects for navigation trees command objects for menu links and tools, tree categories, and toolbar items table object columns, form object fields	Specifies whether the target page should include the Currency Converter tool. In the href URL called when the UI component is clicked, the system passes a parameter called CurrencyConverter and includes the value specified for this setting. If the value is True, the target page includes the Currency Converter tool. If this setting is not included, the value is assumed to be false and the target page does not display the Currency Converter tool. This setting should only be used with Sourcing Central.	True False (default)
Currency Expression table object columns	The name of the currency to convert from. Required for columns whose values are monetary and that can be converted from one currency to another using the Conversion tool and defined exchange rates. The value should be a select clause that returns the value for the Currency attribute for a specific object or relationship, or the actual value. The currency to be converted from should be the currency in which the user entered the data in (the As Entered currency). This setting should only be used with Sourcing Central.	For example, to convert currency data for an RFQ Quotation, the following select clause returns the supplier's currency format. to[Supplier Line Item].attribute[Currency]
Default form object field generic create object fields configurable search form toolbar custom filter	If a field's value is empty or null and this setting is defined, the default value is displayed for the field. Specifies the default value for a toolbar custom filter, and can be a string resource or static text. If the control is configured as a combobox, the Default value must be one of the Range Display Values or Range Values (depending on which setting you used).	The default value you want to display. This can be a string resource key or the actual characters you want to fill in as the default. The wildcard (*) can be used for search criteria fields. emxFramework.Common.default All * emxFramework.CustomControl.Default
Default Category menu objects for navigation trees	Specifies the category that should be selected when the tree first opens or is first inserted into another tree. By default the root node is selected, which means the object's Properties page displays. If the tree contains a category that users frequently want to see, you can use this setting to have it selected instead. Alternatively, you can pass the DefaultCategory parameter to emxTree.jsp. If both are defined, this URL parameter overrides the setting. A tree does not look for default categories defined for its sub-trees (assigned sub-menus).	The name of a category command object that should be selected when the tree first opens. The command object must be assigned to the tree menu object and if it is not, the root node is selected. Default Category=PMCWBS
Delimiter form object fields	When using <code>Input Type = dynamictextarea</code> setting, this setting defines the character that separates values when the field is in View mode. In Edit mode, each value shows on a separate line. A comma is the default delimiter. If you specify any of these characters as the delimiter, a comma is used instead: \$ \ " * ? () >	<any string value> , (default)
Diff Code structure browser columns	When multiple columns in a structure compare Complete Summary Report have different values, this setting defines what to show in the Diff Code column. The code value for the column with the lowest precedence shows in the Diff Code column. Multiple columns can have the same precedence, and if both columns have different values, both code values show in the Diff Code column.	<precedence:code value> 1:Attribute
Display Format	Specifies the number of the date format for any non-	3 - SHORT (12/12/52)

table object columns form object fields configurable search form structure browser pages	editable column or field where the value of the format setting is "date."	2 - MEDIUM (Dec 12, 1952) 1 - LONG (December 12, 1952) 0 - FULL (Tuesday, December 12 1952 AD) Default is set in emxSystem.properties: emxFramework.DateTime.DisplayFormat=MEDIUM. emxSystem.properties uses words, but the Display Format setting uses numbers.
Display Time	Controls whether the time is displayed along with the date for columns and fields whose format is set to date. If no time zone preference is set, then the DateTime is shown in the browser's time zone. The time is shown in terms of GMT+/- hh:mm, (e.g., Saturday, August 21, 2004 12:45:00 PM GMT-04:00). To get the time in a format like EST or PDT, set the time zone preference to a specific zone.	true false Default is set in emxSystem.properties for the property emxFramework.DateTime.DisplayTime
displayMode	Defines in which mode, Edit or View or Both, that this column is visible. When used, the visibility of the column is defined by the current mode and the value of this setting instead of any Access Expression. For example, if the table is in view mode and this setting is Edit, then the column is not visible. This setting applies to columns used in the emxIndentedtable.jsp only; it does not work for columns in flat tables (emxTable.jsp).	Edit View Both (default)
Dynamic Command Function	Defines the method in the JPO specified by the Dynamic Command Program setting that returns a list containing the data structure to build the right-click menu or the categories to add to a tree. In general, the list contains dynamic options based on the user context. This setting is only used with the right-click menu component and the toolbar component.	<JPO Method Name>
toolbar menu objects tree menu objects		
Dynamic Command Program	Defines the JPO invoked from a right-click menu, navigation tree, or toolbar menu component.	<JPO Name>
toolbar menu objects		
Dynamic URL	When enabled, users can enter URLs or mxLink values and the values will display and function as hyperlinks.	enable (default) disable
form and create form object fields table column objects structure browser columns		
Edit Access Mask	Use to control cell-level access in an editable table. This access check is done on all objects in one database call along with getting the column values. If any cell does not have the specified access, then the cell is shown as read only.	Any access mask, for example: modify, connect
table column objects structure browser		
Edit Exclude	Comma-separated list of attributes that will not be included in the webform when opened in Edit mode when a field on the webform has been defined as Field Type = Dynamic Attributes.	For example: attribute_Cost
form object fields		
Editable	Use to indicate whether the column or field is displayed as editable or read only. Only applies for Edit mode. View mode ignores the setting. For forms, default is true. For tables and structure browser, default is false.	true--Users can edit the column or field when shown on the Edit mode form. false--Users cannot edit the column or field when shown on the Edit mode form. The field looks just like it does in View mode except it is never hyperlinked.
table column objects, form object fields, generic create form fields, configurable search form, structure browser		

Effective Date Expression	Used for columns whose values are monetary and that can be converted from one currency to another using defined exchange rates.	For example, for currency data for an RFQ Quotation, the following select clause returns the effectivity date. to[Supplier Line Item].attribute[Effectivity Date]
table object columns	The value should be a select clause that provides the value for the Effectivity Date attribute on a specific object or relationship. The system uses this date to get the currency conversion whose rate period falls within this date. If this setting is not added, the current date is used.	
Expand Function	Specifies the function in the JPO specified in the Expand Program setting that gets the list of child nodes for the tree category.	Name of a method in the Expand Program JPO.
command object for tree category		
Expand Inquiry	Use to configure the tree category so a list of business objects is inserted under it automatically, without users having to first view details for objects. For example, Folder categories are often configured to display all folders without users having to view the folders first. Use this setting to define the list of business objects using an inquiry administrative object. You can also define the list using a JPO with the Expand Program and Expand Function settings. If both an inquiry and JPO are specified, the JPO takes precedence.	Name of an inquiry administrative object that retrieves the business objects to list for the category: TMCFolder TMCSubFolders ENCEBOMList
Expand Program	Use to configure the tree category so a list of business objects is inserted under it automatically, without users having to first view details for objects. For example, Folder categories are often configured to display all folders without users having to view the folders first. Use this setting to define the list of business objects using a JPO. You can also define the list using an inquiry with the Expand Inquiry setting. If both an inquiry and JPO are specified, the JPO takes precedence. This parameter specifies a JPO that contains a method to get the list of objects to insert under the category. The method is specified using the Expand Function setting. Important Note: This feature is recommended primarily for hierarchically organized categories such as Folders and Subfolders. For best performance and consistency of behavior across applications, the preferred method of navigation is to use the standard tree behavior that does not use a dynamic expand. If conditions require the use of this setting, try to restrict its use to categories that will contain relatively few objects.	Name of a JPO program added to the database.
Export	Specifies whether or not column data is exported. Use this setting to change the export value on a column-by-column basis.	true false
table object columns		
structure browser	By default, all column types except programHTMLOutput are exported. To include programHTMLOutput, or exclude other column types from the export, change this setting.	
Field Column Headers	Used in conjunction with the Field Type=Table Holder setting. Specifies the labels for the column headings. The number of labels should be the same as the value for the Field Table Columns setting and should be separated by a comma.	Comma-separated list of column heading labels: Min,Max,Avg
form object field		
create form object field		
Field Row Headers	Used in conjunction with the Field Type=Table Holder setting. Specifies the labels for the row headings. The number of labels should be the same as the value for the Field Table Rows setting and should be separated by a comma.	Comma-separated list of row heading labels: Weight,Volume
form object field		
create form object field		
Field Size	Determines the width of a textbox input type field. The width is given in pixels except when Input Type is	Number of pixels, for example: 30

form object field configurable search form create form object field	textbox or not set. In that case, its value refers to the (integer) number of characters.	20 is the default.
Field Table Columns form object field create form object field	Used in conjunction with the Field Type=Table Holder setting. Defines the number of columns for the table.	2, 3, ...
Field Table Rows form object field create form object field	Used in conjunction with the Field Type=Table Holder setting. Defines the number of rows in the table.	1, 2, ...
Field Type table object columns	Setting used only in edit mode while updating the table data.	<p>basic--The column displays basic information for the business object. Basic information includes name, type, originated, policy, etc. Specifying basic as the field type is only needed when the column is editable. The only editable basic information is: type, name, revision, current, policy, description, owner, vault.</p> <p>attribute--The column displays values for an attribute on the business object, such as Originator or Weight.</p>
Field Type form object field generic crete form object field configurable search form create form object field	<p>This setting is used for several purposes:</p> <p>To indicate that the field's data should be obtained from a program or image instead of an expression.</p> <p>When the field data is obtained from an expression, if the data is basic information or an attribute. The system needs to know whether a field's data is basic information or an attribute in order to update the information correctly. Specifying whether the field is basic or an attribute is only required for fields that will be editable.</p> <p>To indicate the field is a dummy field that defines fields to display in a table or group.</p> <p>These Field Type values that are supported by emxForm.jsp are NOT supported for emxCreate.jsp: emxTable, Table Holder, Group Holder, ClassificationPath, ClassificationAttributes.</p>	<p>program--The values are obtained from a program (JPO). The program and function name are required as settings.</p> <p>programHTMLOutput--Same as program, except the field value output is in XHTML format. Field values are placed in the table cell between <td> and </td> tags. This value ignores other field settings such as Show Type Icon, href, format, and Alternate OID expression.</p> <p>ClassificationPaths--Used only with Library Central. Displays the paths where the object is classified, with a separator between hierarchies. This separator is configurable with the Library Central property string emxLibraryCentral.ClassPathSeparatorString. The default separator is '-----'.</p> <p>ClassificationAttributes--Used only with Library Central and the Multiple Classification Module. Displays the attributes acquired via classification. If the object is classified, it displays the classification name as the heading, then displays a subheading with the name of the attribute group, then attributes and their values acquired from the attribute group.</p> <p>If the same attribute is repeated in another attribute group or in another classification, this field displays a message underneath the field: "This value also appears in another Attribute Group." If a user modifies one attribute that has other occurrences in different attribute groups, this automatically updates all other occurrences of the attribute. If the object is classified but no attributes are acquired via classification, this field does not display anything.</p> <p>Dynamic Attributes--Displays all attribute/value pairs associated with the context object in the properties page.</p> <p>Image--The primary image associated with the business object.</p> <p>basic--Displays basic information for the business</p>

		<p>object: name, type, originated, policy, etc. Use only when the field is editable. The only editable basic information is: type, name, revision, current, policy, description, owner, vault.</p> <p>attribute--The field displays values for an attribute on the business object, such as Originator or Weight.</p> <p>Section Header--Adds a new section heading between the form fields. The setting Section Level determines the heading level.</p> <p>Section Separator--Adds white space to separate fields and sections.</p> <p>Group Holder--Groups the fields under the field in one row. Uses the Group Count setting to determine the number of fields to group. When a field has a Field Type of Group Holder, it serves as a dummy field to define the group and does not appear on the form.</p> <p>Table Holder--Arranges the fields under the field in columns and rows. Table Holder fields serve as dummy fields to define the fields to display in a table.</p> <p>emxTable--Embeds a configurable table in the form. Used in conjunction with the table setting and either the inquiry or program setting.</p>
format		<p>Specifies the type of data in the field. If the Editable setting is true and the field is on an Edit mode form, the system uses these format values to validate the field value. Validation takes place on the client side, before updating the object displayed in the form.</p> <p>To support the date compare logic, whenever the field format is "date", an additional hidden parameter is added to the form with the name assigned to the web form field name suffixed by "_msvalue". If there is a valid display value for the date, the hidden parameter is assigned with the value that is the equivalent of displayed date in milliseconds (calculated from midnight, January 1, 1970).</p> <p>This hidden parameter value can be used by the validation methods to compare two dates.</p> <p>Note: The hidden parameter gets updated only when the out-of-the-box calendar component is used to change the date. If the field type is changed to manual edit, the hidden parameter may not have the updated milliseconds value when the field is manually changed. In this case the validation method can simply ignore the date compare and must depend on the server side validation. The client side validation can be ignored by checking if the field is readonly.</p>
form object field create form object field configurable search form toolbar		<p>date--Uses the tag lib "emxUtil:IzDate" to format the displayed field value.</p> <p>currency</p> <p>numeric--Use if the value must be validated as a number before updating the values. Applicable only in Edit mode.</p> <p>email--Displays the column values as an email address. When a user clicks the email address, the email editor configured in the client is presented.</p> <p>user--Used for create forms (when Editable=false) to add read-only field to support using fullname</p> <p>chooser--Used only on a toolbar with Input Control = textbox when the entered text should be a date or a chooser. The browse button shows after the text box.</p>
format		<p>For custom filter commands, date is the only supported format. When used, the calendar tool shows after the text box.</p>
toolbar commands		<p>date--Displays the column values as a date. Uses the tag lib "emxUtil:IzDate" to format the display.</p>
format		<p>Specifies the format to display the column data.</p>
table object column structure browser		<p>If at least one column has the format set to currency or UOM, the Conversion tool displays in the page toolbar. When a user clicks the tool, the system opens a new window and displays all column data defined with format=currency and UOM to the currency and unit of measure selected in preferences.</p>
		<p>date--Displays the column values as a date. Uses the tag lib "emxUtil:IzDate" to format the display.</p> <p>currency--Displays the column values as currency.</p> <p>UOM--Displays the column values as Unit of Measure and enables the Unit of Measure conversion interface.</p> <p>email--Displays the column values as an email address. When a user clicks the email address, the email editor configured in the client is presented.</p> <p>numeric--Displays the column values as numbers. To perform calculations or graphically analyze the data on a column of string attributes that have numerical values, numeric must be the column type</p>

function	The name of the method to call within the JPO program specified in the program setting. This method within the JPO is used to get the field values if the setting "Field Type" is set to "program" or "programHTMLOutput", or for tables, "checkbox".	The name of a function in the program JPO, such as: getAssignedBuyerDesk getPackageAccess getParentPart getCurrentState
form object field, generic create object field, configurable search form, table object column, structure browser		
Group By table column	Groups rows in the table based on the value in this column. See Group By Column .	true
Group Count form object field	Used in conjunction with the Field Type=Group Holder setting. Specifies the number of fields below the Group Holder field to place in the grouped row.	2, 3, 4, ...
Group Header table object column structure browser	Defines header text to display over several consecutive columns. For example, if you want a group header over three consecutive columns, add this setting to each column and assign the same value for each. To separate grouped columns using a column of white space, add a separator using Column Type=Separator.	Static text or string resource id.
Group Name form objects configurable search form create form object field structure browser	Used for grouping fields in web forms and structure browser. The consecutive fields with same group name are considered a group.	The name of the group.
Help Marker menu objects for navigation trees, command objects for menu links and tools, tree categories, and toolbar items, table object columns, form object field, configurable search form	Specifies the name of the help marker to call for context-sensitive help. In the href URL called when the UI component is clicked, the system passes a parameter called HelpMarker and includes the marker text specified for this setting.	The naming convention for help markers "emxhelp" followed by the object or feature and then the action, for example, emxhelproutecreate and emxhelpprojectedit. The marker is all lowercase with no spaces.
Hide Label form object fields generic create form fields configurable search form	Displays or hides the label for a particular row on a form.	True False
Image command objects for toolbar tools command objects for tree categories configurable Preferences page	Name of the image file to display on toolbar tool commands or next to the label for the tree root node or tree category. The value can be one of the following: Any simple image (gif) file name without any prefix. The system assumes the image is referred from the current directory and prefixes it with "images/". An image file name prefixed with \${COMMON_DIR}. The system looks for the file in the images subdirectory of the directory designated as the common directory. The common directory is defined using the eServiceSuiteFramework.CommonDirectory property in emxSystem.properties. An image file name prefixed with \${SUITE_DIR}. The system looks for the image file in the images subdirectory of the application-specific directory which	iconPerson.gif buttonReports.gif \${COMMON_DIR}/buttonEdit.gif \${SUITE_DIR}/iconCreateECR.gif If no image is specified for a tree root node or category, the system displays the standard no image icon to indicate there is an error in the definition of the tree.

	is defined using the eServiceSuiteSUITENAME.Directory property in emxSystem.properties. For example, the property for Engineering Central is eServiceSuiteEngineeringCentral.Directory = engineeringcentral.	
Image	Use to specify an image file when the field value should be an image only. This setting is required when the Field Type setting is set to image. This file must exist in the application server (not in the database). You can make the image a hyperlink by including a URL in the href parameter.	images/newPart.gif images/EditItem.gif
form object field generic create form fields		
Image	Name of the image used for the menu. This is optional if a label has been defined but required if there is no label.	\${COMMON_DIR}/iconSmallOrganization.gif
toolbar menu object		
Image Size	When image is the Column Type of a table or the Field Type of a web form, this setting defines which size of the primary image associated with the business object should display in the column. The pixel dimensions for these sizes are defined using the emxComponents.Image.SizeType property.	format_mxSmallImage (default) format_mxMediumImage format_mxLargeImage format_mxThumbnailImage
table object column Form object field		
Input Type	Only used for tables or forms in edit mode. Specifies the type of HTML control to display for user input. You can also designate the size of the input boxes to allow for appropriate spacing. This is important for short numeric entry fields. Although multiple choices can be displayed in a webform, only one selection can be saved during edit. To disable the ability to make multiple selections during view of webform, you need to use programHTMLOutput and the html tag that does that. Use the radiobutton or combobox input types for fields that require a single selection. To save multiple selections, a custom JPO needs to be written using the checkbox or listbox input types which might delimit the choices in the attribute using the Studio Customization Toolkit. For toolbar commands, textbox, combobox, and submit are supported. For controls other than toolbar commands, submit is not supported. If you specify checkbox or radiobutton for an attribute with the String datatype, the attribute must be defined with a range or an error occurs when the form is in Edit mode. If using dynamictextarea with a form field, the Delimiter setting also needs to be defined if you want to use a delimiter other than the default (comma).	The setting accepts the values listed below: textbox-This is the default. The field has a single-line box for typing text. textarea-The field has a multi-line box for typing text combobox-The field has a drop-down list of options and users can only select one value. Use comboboxes for attributes that have defined ranges and for attributes whose ranges are determined with a Range Helper URL. listbox-The field has a list of range values. Although users can select more than one item in the list by holding the Ctrl key, it's best to use check boxes for multiple selection fields. radiobutton-For forms only. Shows a radio button next to each range value. checkbox-For forms only. Shows a check box next to each range value. For fields that allow multiple selections, use check boxes instead of a list box. listbox-For forms only. Provides a list of range values. Although users can select more than one item in the list by holding the Ctrl key, only one value will be saved unless a custom JPO is implemented. submit--Submit button configured to send contents of input control to a processing page. dynamictextarea-Allows multiple values to be entered in Edit mode
inquiry		
form object field	Used only for fields defined with Field Type=emxTable. Either this setting or the program setting must be used to define the objects to retrieve. Specifies the inquiry administrative object that should be used to retrieve the business objects to be included in the table.	Names of inquiry administrative object separated by a comma. inquiry=SCSBuyerDesk,SCSBuyerDeskAssigned inquiry=ENCAIIParts, ENCReleasedParts
Label	Use to enter the text that should appear as the label for the field.	--
configurable search form		
Label Function	Specifies the function in the JPO specified in the Label Program setting that gets the value for the root node label.	The name of a method in the Label Program JPO.
menu objects for navigation trees command objects for tree categories		

Label Program	<p>Use to define the label for the root node/tree category using a JPO. This setting specifies a JPO that contains a method to get the value for the label. The method is specified using the Label Function setting.</p> <p>The input for this JPO must include:</p> <ul style="list-style-type: none"> A list of all of the Request Parameters in a HashMap Method Name as a string JPO Program Name as a string <p>The output should be the string that is returned to the tree and used for the label.</p> <p>There are two other ways to define the label for a tree node: using the TreeLabel URL parameter passed to emxTree.jsp and using the Label parameter for the menu object. The URL parameter takes precedence over the other two methods, then the JPO, then the Label parameter.</p> <p>Tree category labels can also be defined using a Label parameter for the command object. The JPO takes precedence over the parameter.</p>	<p>The name of a JPO that has been added to the database.</p> <p>If a JPO is configured for a tree category or root node label and the JPO returns an empty string or null, an exception is thrown.</p>
Level		1,2,3,4,5
toolbar menu object for structure browser	<p>Define the values to show in the Expand filter combo box.</p> <p>The values for this setting can be:</p> <ul style="list-style-type: none"> comma-separated list of positive integers All Specify <p>Text to be processed by the expandProgram JPO.</p> <p>You cannot use custom levels if the structure browser is filtered using relationship/type/direction.</p> <p>The All and Specify reserved keywords control specific levels of expansion.</p> <p>Use the Label setting to define the string resource or static text for the label that shows in front of the expand combo box; use the Registered Suite setting to define the application that defines the menu for the expand filter or the expand Program JPO.</p>	All Specify... Custom expand level (for example, End Item)
Mass Update	Used only for Edit mode. If set to false for the column, the column is not available to the Mass Update feature, but can be edited using the inline editing (one cell at a time). If set to true, the column is used by the Mass Update feature.	true false
Maximum Length	Limits the number of characters that can be entered in a field with Input Type of textbox. If not specified, it uses the HTML default (unlimited).	Number of characters, for example: 30
form object field generic create form field configurable search form		
Maximum Length	Limit the number of characters that are displayed on drop-down menu label. The Title property of the button is used as the Alt text to display the full label.	Any number of characters, such as 25 If this setting is not set, the toolbar displays the entire label.
Message URL Label	Specifies the label to use for URL links that call the tree.	The value can be any select statement, plain text or a string resource ID. For example: \$<attribute[attribute_Title].value> Task Title emxEngineeringCentral.TaskTitle The string resource Id can contain macros also, as in the following example: emxEngineeringCentral.TaskTitle= \$<type>\$<name> \$<revision>: Urgent Task
menu objects for navigation trees	<p>For example, notifications that users have new tasks include URL links that open the Inbox Task tree for the user's task. By default, the URL link label shows the name of the Inbox Task object, which is an auto generated name. But if the route creator entered a name for the task, it is better to show that entered name for the URL link.</p> <p>To have the URL link show the task's entered name, add the Message URL Label setting to the tree menu</p>	

	<p>object for the Inbox Task. The value would be a select macro for the attribute that contains the entered name, in this case the Title attribute.</p> <p>When building the URL link, the system first looks for the Message URL Label setting on the application specific tree menu, for example, ENctype_InboxTask. If not found there, the system looks for the setting in the common tree for that object type. If the common tree does not contain the setting, the system defaults to the object's Type,Name,Revision (for example, Inbox Task IT-0000101).</p>	
Mode	Used for structure browser only.	edit view
toolbar menu object	<p>If defined for a toolbar command, that command is only enabled on the toolbar in the indicated mode (Edit or View). If not defined, the command is enabled in both Edit and View mode.</p> <p>If defined for a toolbar top-level menu, the entire toolbar only shows in the indicated mode and is hidden in the other mode.</p>	
Mouse Over Popup	Enables or disables the mouse over popup DIV that displays the entire contents of the cell.	enable disable (default)
columns used in structure browser	When enabled, the popup shows in both View and Edit mode.	
Name Field	Defines how the name field for the object will display in the form:	autoName
generic create form field	<p>autoName-only the autoName option is available</p> <p>keyin-a textbox is provided for text entry</p> <p>both-both a textbox and the autoName checkbox shown for the name field</p>	keyin (default) both
Nowrap	If the setting is true for any column, then the text displayed in that column will not wrap at any time. The default is false.	true false
table object columns		
Popup Modal	If the setting Target Location is set to popup, the window can be configured as modal or non-modal.	true--the popup window is modal false--the popup window is non-modal If the setting is not specified, the window is modal.
command objects for menu links and tools, toolbar items, table object columns, structure browser,		
Popup Modal	Use to configure whether the popup range helper window is model or non-modal. Only used when the setting Target Location is set to popup.	true--the popup window is modal false--the popup window is non-modal
form object field configurable search form	If a form Component is opened in a dialog in edit mode then the dialog must open as a modal dialog. For example, if a table column is designed to pop up the Form Component in edit mode, then that column must have a setting Popup Modal=true to open the dialog as modal dialog.	If the setting is not specified, the window is modal.
PreExpand	Determines whether to let the JPO program decide whether to add the [+] icon or to simply add the [+] without running the JPO.	<p>true--The Structure Program and Function defined for the Structure command is run to get the list of objects. Then for each object, it runs the same program to determine whether a [+] icon is needed. If the list of objects is greater than 1, a plus [+] is added to the structure node.</p> <p>false (default)--The JPO Program and Function is run to get the list of objects, but the same program and function is not run again for each object. Instead, a plus icon [+] is automatically added for each structure node.</p>
PreExpand Category	Used in conjunction with the Expand Program and Expand Function settings or the Expand Inquiry setting, which inserts a list of objects under the tree category without requiring users to first view details of those	true--Pre-processing is allowed so if there are no objects, the + sign for expanding the list is not shown.
command objects for tree categories		

	<p>objects. Use this setting to determine whether the tree pre-processes the JPO or inquiry to determine if there are any objects to be listed.</p> <p>Pre-processing offers the benefit of determining in advance that there are no business objects in the list so the + sign for expansion is not shown. But pre-processing lowers performance.</p> <p>Administrators who configure this setting as true need to be aware of the performance implications to doing so. While it is nice to only display the + sign when data is present, the tree must actually go and get the data at the time the tree displays to do this. Therefore, if there are many tree categories configured with PreExpand Category set to true, each of those categories will be evaluated when the tree is displayed regardless of whether the user actually intends on viewing data in those categories or not. This potentially unnecessary processing lowers performance.</p>	false (default) --Pre-processing is not allowed so the + sign is displayed even if there are no business objects in the list. The inquiry or JPO is not processes until the user clicks the + sign.
PreExpand Function	Specifies a method in the JPO defined in the Expand Program setting that configures the category for dynamic expand. This method determines whether there are any business objects in the business object list retrieved from the JPO, thereby determining whether the + sign should be displayed.	Name of a method in the Expand Program JPO.
command objects for tree categories	Use of this setting requires a change in ENOVIA Live Collaboration that has not yet been implemented.	
Printer Friendly		True (default) False
menu objects for navigation trees		
command objects for menu links and tools, tree categories, and toolbar items		
table object columns		
form object field	Note that you can also specify the PrinterFriendly parameter in the href URL for JSPs that use it.	
program	Use to specify the name of the JPO program to use to get the field's data when the Field Type setting is set to "program" or "programHTMLOutput".	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
form object field	Also can be used for fields defined with Field Type=emxTable to define the objects in the table. Alternatively, inquiry can be used.	
generic create form field		
configurable search form	Using a JPO to populate field data is recommended only when a select expression cannot be used to obtain the field value.	
program	The name of the JPO program to use to get the column's data. This program is used to get the column values when the setting "Column Type" is set to "program" or "programHTMLOutput" or "checkbox".	emxSCSBuyerDesk emxTMCPackages emxENCParentPart emxCommonTableUtil
table object column	Using a JPO to populate column data is recommended only when a select expression cannot be used to obtain the values.	
Pull Right	Determines whether the drop-down links are displayed in the same level as the menu or pulled right to display the links in the next level. Applies only to second-level drop-down menus.	True (default) False
toolbar menu object		
Range Display Values	Only used with toolbar commands if Input Type=combobox. Holds a comma-separated list of string resources or static text values used to populate the combobox.	emxCustom.Combo.Value1, emxCustom.Combo.Value2, emxCustom.Combo.Value3
toolbar custom filters	If this setting is not specified, the Range Values setting is used.	Value1,Value2,Value3
Range Function	Use to specify the name of the method in the JPO	The name of a function in the Range Program JPO,

toolbar custom filters	specified in the Range Program setting. See the Range Program setting below for more information. Only used with toolbar commands if Input Type=combobox.	such as: getAssignedRange getClassificationRange getPartUOM
Range Program	Use to specify the name of a JPO that contains a method to get the input control value ranges (choices). A range program is used only when the Input Type setting is combobox.	The name of a JPO, such as: ENCParts AEFUtils
Range Values	Only used with toolbar commands if Input Type=combobox. Holds a comma-separated list of values used to populate the combobox.	Value1,Value2,Value3
toolbar custom filters		
Range Function	Use to specify the name of the method in the JPO specified in the Range Program setting. See the Range Program setting below for more information.	The name of a function in the Range Program JPO, such as: getAssignedRange getClassificationRange getPartUOM
form object field configurable search form table object columns structure browser		
Range Program	Use to specify the name of a JPO that contains a method to get the field or column value ranges (choices). A range program is used only when the Input Type setting is combobox, radiobutton, or checkbox for forms, or combobox or popup for tables. If the choices need to be presented on a page in a popup window, for example in a chooser, use the RangeHref parameter instead. The corresponding method which runs to get the values of the column should not have any HTML code.	The name of a JPO, such as: SCSBuyerDeskForm TMCPackagesForm ENCPartForm AEFUtilForm
Read Only Checkbox	For a field defined with the Boolean data type, or as a string with Boolean range values, this setting shows a checkbox instead of the TRUE or FALSE text. In edit mode, the user can check/clear the checkbox unless the Editable=false setting is specified for the field.	true false (default)
form objects		
Registered Suite	The application the UI component belongs to. The system looks for files related to the component in the registered directory for that application, which is specified in emxSystem.properties. Based on the application name, the system passes the following parameters in the href URL: suiteKey SuiteDirectory StringResourceFileId This setting is required for all dynamic UI components.	The value must be set without any spaces, for example, EngineeringCentral or Framework. The value must be set to the suite name as defined in the key eServiceSuites.DisplayedSuites within emxSystem.properties. If the suite name starts with "eServiceSuite" then this prefix can be skipped and assign the remaining text to the setting. For example, if the suite name in emxSystem.properties is "eServiceSuiteEngineeringCentral", then the word "EngineeringCentral", can be assigned as "Registered Suite". In the href URL that is called when the item is clicked, the system passes a parameter called suiteKey and includes the property name that maps to the value specified for the setting. If the Column Type is file, this value must be Components. For subscription commands, use Components.
menu objects for top-level menus and for My Desk and Actions submenus command objects for menu links and tools menu objects for navigation trees table object columns form objects menu objects for toolbars command objects for toolbar items and PowerView tabs structure browser		
Relationship Filter	Optional setting that defines a comma-separated list of relationship select expressions to get the file lists checked into or connected to the Type object.	from [<relationship_ReferenceDocument>].to.id, to [<relationship_ReferenceDocument>].from.id,...,..
table object columns		
Remove Range Blank	Used when the <code>input type</code> is set to <code>combobox</code> to ensure that the field contains a value and is not left blank. If set to True, this setting removes the blank from the combo box list. If a default is not specified, the first value in the list is shown by default.	true false (default)
form object fields configurable search		

form fields		
Required	Use to indicate the value for this field is required. This setting is applicable for editable fields displayed on the Editable mode form.	true--When completing the form, the user must enter a value for the field. The field label appears in red italic text. If the user does not enter a value and clicks Done, a JavaScript message appears that prompts the user to enter a value. false (default)--When completing the Edit form, the user can leave this field blank.
form object field genericcreate form object field		
Revision Filter	Enables or disables the display of the revision filter button.	Enable Disable (default)
tree command objects		
RMB Menu	Specifies the name of the right-click menu to associate with the component.	<admin menu name>
table objects toolbar menu objects	For a type-specific menu, such as type_Part, defines the type-specific right-click menu for that type. For a table column, specifies the right-click menu for that specific column.	
Root Label	Determines the label for the root node(s). You can use static text, such as Search Results, or a string resource id. When using a string resource id, you can use any valid MQL expression, such as: <code>emxFramework.Common.RootLabel = \$<type>\$<name>:Root</code>	Search Results emxFramework.Common.RootLabel
structure browser freeze-pane column	If the expression does not apply to the root object (such as an attribute not associated with that object), the expression is ignored and the default label is used. If this setting is configured for a column not in the freeze pane, it is ignored.	
Row Number	If the column is defined with a Group Name setting, this setting defines whether this column displays in the first or second row of the group. Only 2 rows are permitted for a group.	1 (default) 2
structure browser	Used for merging cells as defined in Merged Cells .	
Row Select	Used only for links on table pages. Specifies whether the JSP specified for the href expects one, at least one, or no rows in the table (Middle Frame) to be selected. If the Row Select setting is set to "single" or "multi", the setting Submit must be set to "true" in order to get the details of the selected item(s) from the table. If the setting is not specified, it is assumed to be none. A JavaScript alert message is displayed on error conditions when the onClick event occurs. The error conditions are explained in the column to the right. For subscriptions, this setting must be set to multi and the Submit setting must be set to true.	single-Appropriate only for links that appear on table pages that have radio buttons or check boxes for each row. The JSP specified for the href expects exactly one row in the table (middle frame) to be selected. If more than one item is selected, a JavaScript alert message is displayed. The default message is set to the value of the following key in emxFrameworkStringResource.properties: emxFramework.Common.PleaseSelectOneItemOnly. Currently, the message is set to "Please Select One Item Only". If no item is selected, then the user will see the following message: "Please select an Item". This message corresponds to this key value in the emxFrameworkStringResource.properties file: emxFramework.Common.PleaseSelectitem multi-Appropriate only for links that appear on table pages that have check boxes or radio buttons for each row. The JSP specified for the href expects at least one row in the table (middle frame) to be checked. If no item is selected, then the user will see the following message: "Please select an Item". This message corresponds to the following key value in the emxFrameworkStringResource.properties file: emxFramework.Common.PleaseSelectitem none (default)-The JSP specified for the href expects no row in the table to be selected. If a row is selected, the JSP does not use it.
Row Span	If this column is defined with a Group Name setting,	1

structure browser	defines how many rows each cell in this column will span. If set to 1, no spanning is done. If set to 2, the column value spans 2 rows. You cannot span more than 2 rows. Used for merging cells as defined in Merged Cells .	2 (default)
Rows	Used when the input type is set to textarea. This setting limits the height of the textarea on the form and specifies the number of visible text lines. If not specified, it uses the HTML default, which is 5.	5 10 s 20
Section Level	Used in conjunction with the Field Type: Section Header setting to define the level of heading.	Two heading levels are available: 1 (default)--Font for heading label is large and a horizontal line is included above the label. 2--Font for heading label is smaller and the heading is in the same gray rectangle as standard fields.
Selectable in Preferences	Only for commands in My Desk submenus. Controls whether the link is available as a preferred Home page. If not set, the system assumes true and the command is listed as a Home page preference. Commands that call pages to display in a popup window, such as an object search command, should not be available as a Home page preference.	True False
command objects for menu links and tools		
Show Alternate Icon	If the column value to be displayed with the icon is different from the current row's object icon, set this setting to true. To get the right alternate icon, the Alternate Type expression setting must be defined with the expression to obtain the object type.	true false
table object column structure browser		
Show Alternate Icon	If the field value needs to be displayed with an icon that is different from the current object type icon, set this setting to true. To get the right alternate icon, the Alternate Type expression setting must be defined with the expression to obtain the object type.	true false (default)
form object field		
Show Clear Button	Adds a Clear hyperlink next to the field. This setting only applies to textarea/textbox input type fields. When the Clear link is clicked, it clears the content of the textbox/textarea input type field.	True False
form object field generic create form field configurable search form		
Show Type Icon	If set to true, the field or column value is displayed along with the object type icon as defined in the emxFramework.smallIcon property in emxSystem.properties. The icon is displayed to the left of the field or column data. If no property is defined for the type's icon, the system looks for a property defined for the parent type, then grandparent type and so on. If no property is defined for any type in the hierarchy, the system uses the default icon specified in the emxFramework.smallIcon.defaultType property.	true false (default)
table object column structure browser generic create form field		
Sort Direction	Used only for fields of type combobox and listbox. The direction to sort the option list.	ascending (default)--Sort a to z or 0 to n. descending--Sort z to a or n to 0. none--The option list is not sorted.
table column objects, structure browser, form object fields		
Sort Program	Specifies the JPO program name that contains a custom sorting algorithm. This setting is used only when the column parameter sortype is assigned to "other".	Name of a JPO added as a program in the database, such as: FindNumberSort CustomSort
table object column		

Sort Range Values	<p>Enables or disables the sorting of range values in combobox or listbox controls. When enabled, the list is sorted based on the datatype and using the direction defined by the Sort Direction setting. When disabled, no sorting is done on the list.</p> <p>In a drop-down, range values populated based on an attribute expression will be sorted based on the attribute's datatype. Range values populated using the Range Program and Range Function settings will be sorted alphanumerically.</p> <p>If the Range Program or Range Function returns numeric or date values, the alphanumeric sort will not be appropriate. The program or function should sort the values in the required order, and this setting should be disabled.</p> <p>This setting does not apply to fields or columns configured for attributes associated with a dimension. Dimension ranges are always sorted as alphanumeric in ascending order.</p> <p>This setting cannot be used in custom JSPs that use the editOptionList taglib. For this specific situation, you can use the sortType attribute for that tag.</p>	enable (default) disable
Sort Type	Determines how the column is sorted.	date integer real AlphaNumericLarger--consider all numeric values as larger than the alphanumeric values AlphaNumericSmaller--consider all numeric values as smaller than the alphanumeric values
table object column structure browser	<p>The MQL <code>sorttype</code> subclause must also be included. Use on its own to sort numeric columns (<code>sorttype = numeric</code>)</p> <p>OR</p> <p>Use in conjunction with the Sort Type setting. Two examples:</p> <p>(<code>sorttype = other AND Sort Type = integer</code>) (<code>sorttype = other AND Sort Type = real</code>)</p>	
Sortable	Controls whether users can sort the table based on data in the table column. If a column is sortable, users can click the column heading to sort the table based on that column.	true (default) false
table object column structure browser		
sortColumnName	Used only for fields defined with Field Type=emxTable. Specifies the column by which the table should be sorted when the page is first loaded. If no column is specified, the rows are listed in the order they are retrieved from the database.	Name of column in the table specified in the table=TABLE_NAME setting.
form object field		
sortDirection	Used only for fields defined with Field Type=emxTable. The direction to sort the rows by.	ascending (default)--Sort a to z or 0 to n. descending--Sort z to a or n to 0.
form object field		
Structure Function	JPO method name to be used for getting the object list.	The name of a method in the JPO specified in the Structure Program setting, such as: getWorkspaceFolders getEBOMs
command objects for structure trees		
Structure Inquiry	Inquiry administrative object name that gets the object list to display within the structure. This is an alternative to the JPO approach and the JPO approach takes precedence.	The name of an inquiry administrative object: SCSWspFolderStructure ENCEBOMList
command objects for structure trees	This setting is not implemented.	
Structure Menu	Specifies the menu administrative object that defines a structure tree for the object type.	The name of a menu administrative object.
menu object for navigation tree		
Structure Program	JPO program that contains the method to be used for getting the object list to display within the structure. The method is specified in the Structure Function setting.	The name of a JPO program added to the database. SCSWorkspaceStructure ENCBOMStructure
command objects for structure trees		

Style Column column in a structure browser	Customizes the style for a column. The value must be a class defined in the dsecUITypeCustom.css. For the example style, the definition in the css file could be: <pre>ColumnBackGroundColor { background-color : rgb(252,186, 186); }</pre>	ColumnBackGroundColor
Style Function column in a structure browser	The method in the Style Program JPO that defines the style per cell in the column.	The name of a function in the Style Program JPO, such as: getStyleInfo
Style Program column in a structure browser	The JPO that contains methods to defines the style per cell in the column.	The name of a JPO, such as: AEFUtilStyleJPO
Submit command objects for toolbar items, including subscription commands	Specifies whether the system should send the object ID(s) (and relationship ID, if applicable) for the current page to the JSP specified in the href parameter. For form pages, the submitted data is the object ID for the business object the page is about. For table pages, the submitted data is the ID for each selected item in the table, which can include object and relationships. In either case, the IDs are sent using the relBusIdList parameter. If the setting is not specified, it is assumed to be false.	true--Submits the IDs to the URL specified in the href parameter. Should be used for actions that depend on selected table rows. false--The URL specified in the href parameter is directly called from the toolbar item and no IDs from the current page are used. Use this value for links that do not require the business object IDs, such links for creating a new object. For toolbar commands in a structure browser, should be true so that the selected items in the structure are posted to the processing page.
table form object field	Specifies the name of a table administrative object to embed in the form. Used only for fields defined with Field Type=emxTable.	Name of table administrative object. For example: table=SCSBuyerDesk table=ENCParts
Target Location command objects for menu and toolbar links and tools table object columns command objects for toolbar items subscription commands structure browser	Controls where the page specified in the href parameter appears. For table columns, this value can be set to any valid frame name, which is available to the table body frame. For toolbar commands, must be listHidden or popup. To specify a tab in a PowerView window, use the administrative command name that configures the tab in the PowerView page. If the specified command name is not defined within the current PowerView, the popup value is used.	content--The page replaces the tree frame. mainframe--The page appears in the main content frame, to the right of the tree frame and tab frames, replacing the page that is already there. (not used for table columns or structure browser) popup--Page appears in new window. The window modality is set using the Popup Modal setting. _top--Page replaces the entire body of the browser window. topFrame--The page replaces the banner frame. (not used for table columns or structure browser) hiddenFrame--The frame called hiddenFrame is part of the top level Navigation window frameset and can be used to submit the Table frame and carry out background processing. This frame is not available for popup windows though so listHidden is a better frame to use. listHidden--A hidden frame within the table frameset. Use this frame for Target Location for any processing in the context of a table. <command name>--The name of a command configured as the target tab in a PowerView page.
Target Location form object field configurable search form	Controls where the page specified in the href parameter appears when a user clicks the hyperlinked data. This value can be set to any valid frame name that is available to the form body frame.	popup--Page appears in a new window. The window modality can be set with the Popup Modal setting. Popup is the default value and is used if the setting is not included or if the named frame cannot be found. content--The page replaces the content frame. mainFrame--Page appears in the frame that includes the content and menu frames. _top--Page replaces the entire body of the browser window.

Target Location	This must be assigned to "searchContent."	searchContent
Search JSP page		
Tip Page menu objects for navigation trees command objects for menu links and tools, tree categories, and toolbar items table object columns form object field	Specifies whether the target page should include the Tip Page tool and the URL that should be called when the tool is clicked. In the href URL called when the UI component is clicked, the system passes a parameter called TipPage and includes the URL specified for this setting. If this setting is not included, the target page does not display the Tip Page tool.	Name of a custom html or JSP page, including any path. The starting point for the directory reference is the content directory. For example, if you want to call an html file in ematrix/doc/customcentral and the content directory is ematrix/customcentral, you would add this parameter to the table.jsp: TipPage=../doc/customcentral/tippage.html
Tree Scope ID menu objects for tree nodes, menu objects for navigation trees	Passes the current tree's objectId to all its categories, to subtrees inserted into the tree, and to categories in the subtrees, to the nth level, in the parameter name specified as the value for this setting. This setting is valid only for tree menu objects and not for tree category command objects. For example, suppose you want to pass the object ID for the current workspace to all categories and subtrees in the workspace tree. You want to pass the object ID using a parameter called workspaceId. To do this, you would add the Tree Scope ID setting to the menu administrative object for workspace trees (which is called type_Project due to administrative object renaming) and enter "workspaceId" as the value. The parameter and value "workspace Id=OBJECTID OF THE WORKSPACE will be available to all the categories and subtrees of the current workspace to the nth level until another workspace object gets inserted or until any explicit URL parameter with the same name "workspaceId" is passed to the tree.	The name of the parameter you want to use to pass the object ID. For example: workspaceId projectId partId You can also pass specific parameters from the tree menu object to its categories using the AppendParameter parameter for the emxTree.jsp.
Type Ahead Mapping form object field, table column	When Type Ahead is configured on the field/column (by setting the RangeHref to emxFullSearch.jsp or using a predefined Type Ahead Chooser), this setting maps that field to the corresponding indexed field in config.xml. See About Automatic Type Ahead .	NAME LASTNAME,FIRSTNAME,USERNAME
Type Ahead Validate form object field, table column	When Type Ahead is configured on the field/column, this setting determines if the application should restrict the user to selecting one of the suggested values (if true), or if the user can enter any data in the field without BPS validating that data (if false).	true false (default)
Type Icon Function tree menu objects web forms tables structure browsers	Specifies the function in the JPO specified in the Type Icon Program setting that retrieves an icon to show in addition to the icon defined by the emxFramework.smallIcon.type system property.	Function Name
Type Icon Program tree menu objects web forms tables structure browsers	Defines the program that contains the function specified using the Type Icon Function setting.	JPO Name
TypeAhead configurable search form field, structure browser mass update	Enables type ahead. For mass update in a structure browser, default is true.	true false
TypeAhead Program configurable search form field, structure	Name of the JPO called to retrieve a list of possible values.	--

browser mass update		
TypeAhead Function	The name of the JPO method.	--
configurable search form field, structure browser mass update		
TypeAhead Character Count	Overrides the emxFramework.TypeAhead.RunProgram.CharacterCount system property.	2 (default) 5
configurable search form field		
TypeAhead Saved Values Limit	Overrides the emxFramework.TypeAhead.SavedValuesLimit system property.	10 (default)
configurable search form field		
UOM Expression	The name of the Unit of Measure to convert from. Required for columns whose values are measurements that can be converted from English units to Metric units or vice versa.	For example, for measurement data for RFQ Quotations, the following select clause returns the supplier's unit of measure format. <code>to[Supplier Line Item].attribute[Unit of Measure]</code>
table object columns	The value can be a select clause that returns the value for the Unit of Measure attribute for a specific object or relationship or the actual value. The unit of measure to convert from should be the unit of measure in which the user entered the data (the As Entered unit of measure). This setting should only be used with Sourcing Central.	
Update Function	Specifies a method name in the JPO given in the Update Program setting.	The name of a function in the Update Program JPO, such as: <code>setAssignedBuyerDesk</code> <code>setPackage Access</code> <code>setPartClassification</code>
form fields, generic create form fields, table object columns, structure browser	Generic Create Form: This setting can not be used with the required fields: Type, Name, Policy, Vault, Owner.	
Update Program	Specifies a JPO that contains a method to set the field value when the field is displayed on an Edit mode form and when the user clicks Done. This program is used only when the Field Type setting is program or programHTMLOutput.	The name of a JPO, such as: <code>SCSBuyerDeskForm</code> <code>TMCPackagesForm</code> <code>ENCPartForm</code> <code>AEFUtilForm</code>
form fields generic create formfields table object columns structure browser	Using an update program is recommended only when the Field Type is not attribute or basic. Generic Create Form: This setting can not be used with the required fields: Type, Name, Policy, Vault, Owner.	
Update Program Arguments	Comma-separated list of arguments to be passed to the JPO specified by the Update Program setting.	Relationship=relationship_EBOM,ConnectFrom=true
structure browser mass update		
Validate	Specifies the method to be invoked for validating any cell.	The name of a method, such as: <code>checkUniqueName</code>
table column object, form object field, generic create form field, structure browser		
Validate Type	This setting is used to validate any column or field value with any specific characters defined in emxSystem.properties.	Basic or Restricted--The column values are validated against the value of emxFramework.Javascript.BadChars. Name--The column values are validated against the value of emxFramework.Javascript.NameBadChars
table, column object, form field, generic create form field, structure browser		
Vertical Group Name	Used for grouping fields vertically in create web forms. The fields with the same Vertical Group Name value will be considered a group.	The name of the vertical group.
create form object		

field		
View Exclude	Comma-separated list of attributes that will not be included in the webform when opened in View mode when a field on the webform has been defined as Field Type = Dynamic Attributes.	For example: attribute_Cost
form field object		
Width	Number of pixels to define the column width.	Column width in pixels: 150
structure browser	When this setting is not defined, by default the width is calculated based on the header text length, so that the header text is not truncated.	
width	Specifies the width of the filter toolbar control in pixels. Use this setting if you need to display many controls in a small area.	100
toolbar custom filter		
Window Height	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. Use to define the height of the popup chooser window or window opened from any href link in a form field. This value is used only when the setting for Target Location is set to "popup".	Number of pixels, such as 700: 400 600(default) 800
command objects for menu links and tools command objects for toolbar items, form object field, structure browser, configurable search form field	Deprecated. This setting is overridden by the emxFramework.Popup.Default property set in emxSystem.properties. Use to define the width of the popup chooser window or window opened from any href link in a form field. This value is used only when the setting for Target Location is set to "popup".	Number of pixels, such as 700.: 400 600(default) 800
Window Width		

URL Parameters

This table lists the parameters that you can specify for configurable JSPs.

Parameter	Description	Accepted Input Values
Accepted by JSP		
actionMenuItemName emxLifecycle.jsp	The name of the menu.	Name of any menu object that represents a menu. The framework installs with a command object called AEFLifecycleMenu that includes a Promote and Demote link.
addJPO emxIndentedTable.jsp	The JPO name and method name to execute when the user adds an empty row to the structure browser for creating a new object. If both the <code>addJPO</code> and <code>applyURL</code> parameters are defined, the <code>addJPO</code> URL parameter is ignored.	JPOName:methodName
appendColumns emxTable.jsp, emxTableEdit.jsp, emxIndentedTable.jsp	The name of the table that contains defined columns that you want to include in the table being defined.	
appendFields emxForm.jsp, emxCreate.jsp	The name of the web form that contains defined fields that you want to include in the form being defined.	
AppendParameters emxTree.jsp	Passes common parameters to the tree categories in the tree. To use the parameter, set it to tree and then include the parameters to pass. Parameters added to <code>emxTree.jsp</code> that are not with <code>AppendParameters</code> are passed only to the root tree node and not to categories. You can also use the Tree Scope ID setting for the tree menu object to pass the current tree's object ID to all categories and subtrees.	To make Param1 and Param2 available to the children categories: <code><a href="..//common/emxTree.jsp?objectId=<%=sPartId%>&mode=insert&jsTreeID=<%= jsTreeID %>&AppendParameters=true&Param1=Value1&Param2=Value2" class="object" target="content"><%=sPartName%></code> To make ProjectId and Workspace available to the tree categories: <code><a href="..//common/emxTree.jsp?objectId=<%=sPartId%>&mode=insert&jsTreeID=<%= jsTreeID %>&AppendParameters=true&ProjectId=<%=sProjId%>&Workspace=<%=sWorkspace%>" class="object" target="content"><%=sPartName%></code> To make ProjectId and Workspace available only to the root tree node and not to the tree categories: <code><a href="..//common/emxTree.jsp?objectId=<%=sPartId%>&mode=insert&jsTreeID=<%= jsTreeID %>&&ProjectId=<%=sProjId%>&Workspace=<%=sWorkspace%>" class="object" target="content"><%=sPartName%></code>
applyURL emxIndentedTable.jsp	Specifies the URL string to be submitted when a user clicks the Apply button after making changes to the structure browser in edit mode. If both the <code>addJPO</code> and <code>applyURL</code> parameters are defined, the <code>addJPO</code> URL parameter is ignored.	
autoFilter emxTable.jsp emxIndentedTable.jsp	Determines whether the filter tool  displays in the toolbar, and overrides the Auto Filter setting on a table or structure browser column.	true (default)--The Filter tool displays in the page toolbar. false--The Filter tool does not display in the page toolbar.
calculations emxIndentedTable.jsp, emxTable.jsp	Controls the display of the  toolbar button. When true, this icon only displays on the toolbar if the structure browser/table contains at least one column defined as numeric.	true--Sigma icon shows if the structure browser contains at least 1 numeric column (default for tables) false--Sigma icon does not show on the toolbar (default for structure browser)
callbackFunction emxFullSearch.jsp	Optional. The name of the function invoked when a user clicks the Submit button on the search form. The <code>emx{Suite}.FullText.CallbackInclude</code> property in <code>emxSystem.properties</code> defines the name of the file that includes this function (where Suite is the application name).	<string>
CancelButton emxTable.jsp, emxFormEditDisplay.jsp	Displays the Cancel link in the table footer frame. When a user clicks Cancel, the whole browser window gets closed. So the Cancel link can only be used for a table page that is opened in new window. The label for the Cancel link can be configured using the <code>CancelLabel</code> parameter. If <code>CancelLabel</code> is not passed in, the default label "Cancel" is used.	true--Shows the Cancel link. Use only for tables displayed in popup windows. false (Default)--Does not show the link.
CancelLabel	Defines the label for the Cancel link, if you want something other than "Cancel".	Any static text or string resource id.

emxTable.jsp, emxIndentedTable.jsp, emxFormEditDisplay.jsp, emxFullSearch.jsp		CancelLabel=emxEngineeringCentral.Common.Cancel
cancelProcessJPO	The JPO to use when a user clicks the Cancel or close window button on a table or form component.	<JPOName>:<MethodName>
emxTableEdit.jsp emxFormEdit.jsp emxIndentedTable.jsp		
cancelProcessURL	The JSP called when a user clicks the Cancel button or closes the table or form page. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.	\${SUITE_DIR}/emxCustomCancelProcess.jsp or ..<ApplicationDirectory>/emxCustomCancelProcess.jsp Example: ../engineeringcentral/emxCustomCancelProcess.jsp
chart	Specifies whether the chart options icon is shown in the table toolbar.	true (default)--Show the chart options icon in the table toolbar if the table has one or more numerical columns. No icon is shown if the table does not have any numerical columns. false--Do not show the chart options icon in the table toolbar, even if the table has numerical columns.
emxTable.jsp		
chartType	The type of the chart to be drawn.	BarChart StackBarChart PieChart LineChart
emxChart.jsp		
chartTitle	The title for the chart.	
emxChart.jsp		
CommandName	Use along with the MenuName parameter to define the page that should be displayed in the Content frame of the Navigator page when users log in. The MenuName parameter defines the name of a menu object that represents a submenu and the CommandName parameter defines the name of a command object from which the href URL is to be obtained. Note that the links in the Actions tab submenus cannot be used for display in the content frame.	The name of a command object that is assigned to a menu object that represents an application submenu on the My Desk menu. For example: emxNavigator.jsp?MenuName=TMCMYDes&CommandName=TMCRoutesMyDesk Where TMCMYDes is the name of the menu object for the Team Central submenu of the My Desk menu and TMCRoutesMyDesk is the name of the command object for the Routes link. This command object contains an href parameter that points to the JSP for the Routes page.
compareBy	One or more column names from the table. The value can be a comma-separated list.	Quantity,Find Number
emxStructureCompare.jsp		
connectionProgram	Defines the JPO:method to connect objects in a given structure hierarchy. You only need to write and specify a connectionProgram if you need to implement custom logic for connecting objects.	emxPart:connectEBOMParts emxDocument:connectFolders
emxIndentedTable.jsp emxStructureCompare.jsp	For the structure browser, The JPO:method used for operations such as resequence, Copy, Paste, and Cut & Paste to make the needed connection between child and parent objects. This parameter is only used to implement custom logic	
ContentPage	Defines the JSP that should appear in the Content frame (called the Home page in the user interface) when the Navigator page is called.	Any JSP plus parameters: emxNavigator.jsp?ContentPage=../engineeringcentral/emxengchginboxFrameset.jsp Currently, the URL assigned to the ContentPage parameter can contain only one parameter/value pair as part of the URL. Any additional parameters are ignored.
emxNavigator.jsp		
customize	Enables (true) or disables (false) the ability for users to create customized versions of the page. This parameter overrides the emxFramework.UITable.Customization system property, and has no effect on other table or structure browser pages. If not provided, the value specified for <code>emxFramework.UITable.Customization</code> is used. All access controls defined for the system table are retained in the customized table. If a column does not have a label or alt value defined (and is not a file, checkbox, icon, image, or separator), the column name shows in the user's selection list, however, this name cannot be internationalized.	true false
emxTable.jsp emxTableEdit.jsp emxIndentedTable.jsp		
default	For each field in the form-based view, you can pass default values. The search page uses these values to load the initial search results, and users can change these values as needed.	default=POLICY=policy_ECPart,policy_DevelopmentPart
emxFullSearch.jsp		

DefaultCategory	Specifies the category that should be selected when the tree first opens or is first inserted into another tree. By default, the root node is selected, which means the object's Properties page displays. If the tree contains a category that users frequently want to see, you can use this parameter to have it selected instead. Alternatively, you can use the Default Category setting for the tree's menu object. If both are defined, this URL parameter overrides the setting. Note that a tree does not look for default categories defined for its sub-trees (assigned sub-menus).	The name of a category command object that should be selected when the tree first opens. The command object must be assigned to the tree menu object and if it is not, the root node is selected. DefaultCategory=PMCWBS
defaultSearch	The URL parameter can be passed with the name of the Search command. The JSP page configured for this command is used for displaying the search page when opening the Search window.	SPCSCOSearch
emxSearch.jsp or any search JSP page	If the logged in user does not have access to the designated "defaultSearch" command, the first available and accessible search command will be displayed. If no "defaultSearch" is passed in, then by default the General TNRV search page will be displayed.	TMCSearch Default is the first available command in search toolbar AEFSearchToolbar. The first available command in the framework install will be the general search command AEFGeneralSearch.
direction	Structure browser: Direction to be used for expanding the object. When one or more relationship name is passed in, the object will be expanded for the given relationship(s) using the specified direction. Generic Create form: If a relationship is passed, the direction parameter specifies the direction of the relationship from the perspective of the object being created. If from, the relationship would be FROM the passed object TO the newly-created object. If to, the relationship would be FROM the newly-created object TO the passed object.	Structure Browser: to from both (default) Generic Create form: From (default) To
directionFilter	Used to turn ON or OFF the direction filter shown in the header. By default the parameter is false and the direction filter is hidden. The parameter must be passed explicitly as true to show the filter.	true false (default)
disableSorting	By default all the columns are sortable, unless a column has a setting 'Sortable = false'. If the user clicks on the header of the sortable column, then the objects are sorted based on that column.	true false (default)
emxTable.jsp	To prevent users from sorting columns, pass the parameter 'disableSorting=true' to emxTable.jsp.	
draw3D	Indicates whether the chart is drawn in 3D or not.	true--Draws the chart in 3D false--Draws the chart in normal mode (default)
emxChart.jsp		
editable	setting that tells the search component whether to make the type field editable or not. A value of true will make the text box editable. A value of false will make it read-only. If this parameter is not passed into the search component, then the default will be false. The type field will only be set to true if the browser language setting is set to "English". If this value is set to true and the browser language is NOT set to English, then this setting is ignored and reverts to false.	true false
editLink	Use to display the Edit toolbar item on the table or form page. The Edit link is automatically configured to call the editable version of the current table or form using emxTableEdit.jsp or the form edit.jsp.	true false (default)
emxTable.jsp	Use to have the system include a default Edit toolbar item in the View Form header.	emxForm.jsp?form+ENCPart&editLink=true
emxForm.jsp		
emxIndentedTable.jsp	For structure browser: when true, the Mode menu and disabled Edit menu (AEFSBEditActions) is included in the toolbar.	
editRelationship	Defines a comma-separated list of relationships that can be used in conjunction with the cut and paste edit commands. The user can only cut or paste rows that are connected to the parent object using one of the defined relationships.	relationship_AffectedItems, relationship_EBOM
emxIndentedTable.jsp		
emxSuiteDirectory	The name of the directory for the application under ematrix.	engineeringcentral
configurable pages, many standard pages, most custom pages	Not explicitly passed: passed by the Registered Suite parameter.	
enableSearchButton	The <code>emxFramework.FullTextSearch.EnableSearchButton</code> property defines globally whether the Search button is enabled prior to the user entering criteria. This parameter overrides that property for the specific page. The default value for this parameter is the value set for this property in <code>emxSystem.properties</code> .	true false
emxFullSearch.jsp		

excludeOIDprogram	The JPO program:method that returns a list of Object IDs (OIDs) that should not display in the result list. This program:method provides an additional level of filtering.	JPO program:name emxECR:getAffectedItemsAlreadyConnected
ExclusionList	Use to define the types to exclude from the top-level list of types that display when the Type Chooser opens. The inclusion list and the exclusion list cannot both be passed as parameters. If they are, an error message is displayed. If neither is specified, all top-level types are listed.	a properties file key If the SuiteKey parameter is passed in, the system looks for the key in the application-specific properties file (for example, emxEngineeringCentral.properties). If the property is not application specific, the system looks in emxSystem.properties. For example, if emxEngineeringCentral.properties contains this property: eServiceEngineeringCentral.Types=type_Part,type_ECR,type_Sketch The parameter to pass would be <code>ExclusionList=eServiceEngineeringCentral.Types</code> a comma delimited list of symbolic names for the types to exclude: <code>ExclusionList=type_Part,type_ECR</code>
ExclusionList	Means that the type(s) that is passed in will not be displayed in the type chooser. All other types will be displayed and users can select the types they wish to search for from the type chooser. The ExclusionList can either be a property value key or a list of types.	ExclusionList =type_part, type_ECO ExclusionList= emxFramework.GenericSearch.ExclusionList
any search JSP page		
expandLevelFilter	Enable or disable the Expand filter for the structure browser page. This parameter overrides the <code>expandLevelFilterMenu</code> URL parameter.	true (default) false
emxIndentedTable.jsp		
expandLevelFilterMenu	Specify the menu used to show the Expand level filter.	Administrative menu name AEFFreezePaneExpandLevelFilter (default)
emxIndentedTable.jsp		
expandProgram	Use to pass the name of the program to use for expanding the object. This parameter value should be assigned to the JPO name and the method name separated by a colon.	<JPO Name:method Name> emxPart:getEBOM emxDocument:getFolders
emxIndentedTable.jsp		
emxStructureCompare.jsp		
expandProgramMenu	Can be assigned to an administrative menu or a command name, which contains the definitions of the JPO, method name and label. When command is passed in: Command settings <code>program</code> and <code>function</code> are used for getting the JPO name and method name for expanding the objects. The List Filter will not be shown, as there is only one JPO available to expand. When Menu is passed in: Commands connected to this menu are used for listing the options in the List Filter combo box. The label for each item is obtained from the command's label. The <code>program</code> and <code>function</code> settings on individual commands are used as the JPO method for expanding the objects, upon selecting the options from the List Filter. By default, the first command in the list is used for expanding the objects. The commands connected to the menu honor all the access settings supported by the configurable component (such as Access Mask, Access Expression, and Access Program).	Administrative menu name Administrative command name For example: ENCBOMLists (menu) PMCFolderLists(menu) ENCBOMList (command)
emxIndentedTable.jsp		
Export		true (Default)--The Export tool displays, allowing users to export the table data. false--The tool does not display.
emxTable.jsp	Shows or hides the Export  tool on the page toolbar. When users click the tool, the system exports the table data to a file in the user's preferred format in their preferred format: CSV, HTML, or Text. Users choose their preferred format using the Preferences tool in the global toolbar.	
emxIndentedTable.jsp	If the user chooses CSV or Text for their export format, the export does not include Icon column type and programHTMLOutput columns. To get the column values exported when the column type is "programHTMLOutput," you must also set "Export= true" for the column.	
field	A colon-separated list of indices and the default/allowed values for the initial search results. See Initial Search Criteria .	Type=type_Part:Originator='Test Everything':Policy=policy_P1,policy_P2:LastRevision=true
emxFullSearch.jsp		
fieldNameActual	Required for the Type Chooser. Use to specify the field on the form page to populate the type(s) the user selects in the Type Chooser. The system populates the specified field with the actual AEF type names, as stored in the database, so the field must be a hidden field. When calling the Type Chooser using a RangeHref on a configurable Form page, the hidden field is created automatically and has the same name as the	The name of a hidden field on the form page. For example, if the form page contains a field defined as follows: <code><input type="hidden" name="txtSelectedTypeName" value=""></code>
emxTypeChooser.jsp		
emxFullSearch.jsp		

	<p>RangeHref field.</p> <p>Other pages that need to get the type name selected by the user, for example to perform a search, should get the type name from this field instead of the fieldNameDisplay.</p>	<p>This parameter should be passed to emxTypeChooser.jsp:</p> <p><code>fieldNameActual=txtSelectedType</code></p>
fieldSeparator	Defines the character to use to separate fields in the URL string. This parameter overrides the value set using the <code>emxFramework.FullTextSearch.FieldSeparator</code> property defined in <code>emxSystem.properties</code> .	<code>^sep^</code>
emxFullSearch.jsp	<p>Do not use any of these values:</p> <p>value of the <code>emxFramework.FullTextSearch.RefinementSeparator</code> property (these 2 properties must have different values)</p> <pre># ! & % (any text that begins with the percent symbol, such as %3A, %3D, %2C) [or] {or } ? ' "</pre>	
filterColumnPosition	<p>Optional. Defines where dynamically-generated columns based on a user's attribute selection will be inserted in the results table.</p> <p>The value must be non-negative and cannot exceed the number of columns in the results table. Or, you can use last to indicate that the column(s) should be inserted after the last column.</p> <p>If no parameter is passed, the columns are inserted after the structure browser pane divider.</p>	A positive integer or the reserved word <code>last</code> (case insensitive)
fieldNameDisplay	<p>Required for the Type Chooser. Use to specify the text field on the form page to populate the type(s) the user selected in the Type Chooser and display these types to the user. When calling the Type Chooser using a RangeHref on a configurable Form page, the display field is created automatically and has the same name as the RangeHref field with "Display" appended to it.</p> <p>This field contains the internationalized version of the selected type list.</p> <p>If <code>fieldNameActual</code> is used and the actual value and display value are different, this parameter defines the name of the field where the display value should be returned.</p>	<p>The name of a non-hidden field on the form page. For example, if the form page contains a field defined as follows:</p> <pre><input type="text" name="txtType" value="" onClick="showTypeChooser()"></pre> <p>This parameter should be passed to emxTypeChooser.jsp:</p> <p><code>fieldNameDisplay=txtType</code></p>
FilterFramePage	Specifies the name of the JSP that will be used in the custom filter frame, which is between the header and body. You can use the directory macros to refer the location of the page. The path of the page is relative from the Apps/Framework/VERSION/commonUI directory.	<p><code> \${SUITE_DIR}/emxTeamProjectTableFilter.jsp?</code></p> <p><code> \${ROOT_DIR}/emxPartECOSatusFiletr.jsp?</code></p> <p><code> \${COMMON_DIR}/emxCommonStateFilter.jsp?</code></p> <p><code> emxCommonStateFilter.jsp</code></p>
FilterFrameSize	Controls the size of the filter frame in pixels. The value should be at least 40, which is the default.	<p>40 (default)</p> <p>80</p> <p>160</p>
findMxLink	When true, show the mxLink icon/command button on the toolbar, which opens a search dialog box. The icon only appears on toolbars on a Create or Edit page; if passed to a View page it does not show on the toolbar.	true
generic search, emxForm.jsp, emxCreate.jsp, emxTable.jsp, emxTableEdit.jsp, emxIndentedTable.jsp	<p>Default is false for toolbar.</p> <p>When toolbar is for a form, table, or structure browser, default is true.</p>	false
form	Specifies the web form administrative object used for presenting the form page.	<p>Name of web form administrative object.</p> <p><code>emxForm.jsp?form=ENCPart</code></p> <p><code>emxCreate.jsp?form=SCSBuyerDesk</code></p>
emxForm.jsp, emxCreate.jsp, emxFormEditDisplay.jsp		
formHeader	Use to define the content of the heading that appears at the top of the form page, in the header frame. The value can be either a string resource id or the label itself. The text in the label and string resource ID can contain any valid select expression.	<p><code>emxForm.jsp?form=ENCPart&formHeader=Properties</code></p> <p><code>emxForm.jsp?form=ENCPart&formHeader=Properties:\$<type>\$<name></code></p> <p><code>emxForm.jsp?form=ENCPart&formHeader=emxQuoteCentral.AssignedPackages.AssignedPackages</code></p>

formInclusionList	Comma-separated list to limit the list of indexed attributes to display on a form-based search. This parameter does not affect navigation-based searches.	Name,Description,Policy,attribute_title
emxFullSearch.jsp		
formName	Use to specify the name of the form that holds the field specified in fieldNameDisplay. If this parameter is not passed in, the Type Chooser looks for the field name on the first form of the form page. If the field name is not found in the first form, then an error message will be presented.	The name of a form on the form page. For example, if the form page contains a form defined as follows: <pre><form name="searchPage" method="post" action="somejsp.jsp"></pre> This parameter should be passed to emxTypeChooser.jsp: <code>formName=searchPage</code>
frameName	Use to specify the name of the frame that contains the form on the form page. Needed for pages that contain forms in multiple frames.	The name of a frame on the form page.
emxTypeChooser.jsp emsFullSearch.jsp	If using form-based search, this parameter provides the name of the frame that contains the form.	
freezePane	Used to configure which column(s) should be displayed as the freeze pane column in the structure browser. If this parameter is not passed in, the first column in the table is used as Freeze Pane column. A comma-separated list can be provided, and the columns are displayed in the order listed. If all of the table columns are passed, the last column listed for this parameter displays in the scroll pane.	Column name in the table Name Title Name,Title,Description
genericDelete	When true, shows the Delete action (AEFGenericDelete command) on the toolbar for the search page.	true
emxFullSearch.jsp	When false, the Delete action is not included in the toolbar.	false (default)
header	The content of the heading that appears at the top of the table page.	Any alphanumeric text or a string resource ID.
emxTable.jsp emxIndentedTable.jsp emxChart.jsp emxCreate.jsp	Generic Create Form: The header displayed in the header frame of the form. The value can be a string resource ID or the label itself.	header=Buyer Desk header=emxQuoteCentral.AssignedPackages.AssignedPackages
Header	The text to use for the header of the History page. This parameter can either be a string resource ID or a mixture of macros and text or simply text.	\$<type> \$<name> \$<revision> emxFrameworkStringResource.Common.HistoryPageHeading Object History
header	The content of the heading that appears at the top of the portal page.	Any alphanumeric text or a string resource ID. header=SummaryView header=emxSpecificationCentral.SCOSummaryView.SummaryView Text string or string resource, such as: Lifecycle emxFramework.Lifecycle.LifeCyclePageHeading
headerRepeat	Specifies how often the header row should be repeated. A repeating header row makes it easier for users to identify the content of each column when scrolling through a long table. If the parameter is not used, the header row repeats every 15 rows (or whatever number is assigned to the emxSystem.properties key emxNavigator.UITableView.HeaderRepeat).	Integer For example, 11 would mean the header row is repeated every 11 rows. To turn off headerRepeat, pass in 0 as the value for the parameter.
HelpMarker	Specifies the name of the help marker to call for context-sensitive help.	String The naming convention for help markers "emxhelp" followed by the object or feature and then the action, for example, emxhelproutecreate and emxhelpprojectedit. The marker is all lowercase with no spaces.
emxTable.jsp, emxForm.jsp, emxCreate.jsp, emxIndentedTable.jsp, emxPortal.jsp, emxFormEditDisplay.jsp and most standard pages		
hideHeader	The header includes the search text box, and the Search and Reset buttons. To hide these items, set this parameter to true.	true false (default)
emxFullSearch.jsp		
hideRootSelection	Shows or hides the check box for the root node when the structure browser is configured for a single root node and the <code>selection</code> URL parameter is set to multiple (ignored if selection is set to single or none or the structure browser includes multiple root nodes).	true false (default)
emxIndentedTable.jsp		
hideToolbar	Shows or hides the toolbar in the search results section of the search page. To hide	true

emxFullSearch.jsp	the toolbar, set this parameter to true.	false (default)
HistoryMode	Determines whether the entire revision chain or just the current revision history should be displayed. For example, a part may have REV A, Rev B and Rev C. If we are reviewing REV B and the HistoryMode is set to "CurrentRevision", only the history for REV B is presented. If HistoryMode is set to "AllRevisions", then the history for REV A, B, C is presented.	AllRevisions--The page displays the entire revision history of the object. CurrentRevision (default)--The page displays the history for the particular revision that is being reviewed.
InclusionList	Use to define the top-level list of types to display when the Type Chooser opens. Note that these types do not have to be top-level types (types with no parents), they will just be listed in the top level in the chooser. The inclusion list and the exclusion list cannot both be passed as parameters. If they are, an error message is displayed. If neither is specified, all top-level types are listed.	a properties file key If the SuiteKey parameter is passed in, the system looks for the key in the application-specific properties file (for example, emxEngineeringCentral.properties). If the property is not application specific, the system looks in emxSystem.properties. For example, if emxEngineeringCentral.properties contains this property: eServiceEngineeringCentral.Types=type_Part,type_ECR,type_Sketch The parameter to pass would be <code>InclusionList=eServiceEngineeringCentral.Types</code> a comma delimited list of symbolic names for the types to include: <code>InclusionList=type_Part,type_ECR</code>
InclusionList any search JSP page	Means that the type(s) passed in will override the list of types defined in the properties file. The type or list of types will be displayed in the type chooser and users can select the types they wish to search for. The InclusionList can either be a property value key or a list of types.	InclusionList =type_part, type_ECO InclusionList= emxFramework.GenericSearch.types
inquiry emxTable.jsp emxChart.jsp	Used when there is no program parameter passed in to emxTable.jsp or emxTableEdit.jsp. Specifies the inquiry administrative object that should be used to retrieve the business objects to be included in the table and any inquiries for the table page filter list. emxTable.jsp uses the first inquiry object to display the list in the table when the page is first loaded. It uses all inquiry objects for the table filter list in the header. emxTableEdit.jsp does not support multiple inquiries. If multiple inquiries are passed with the ',' separator, only the first one will be used and rest are ignored. If there is only one inquiry, the filter list in the header is not displayed. Currently, to build the filter list, you must use either JPOs or Inquiry objects. You cannot use both. For charts:Assigned to one inquiry administrative object name. The inquiry will be used to get the objects used to draw the chart.	For emxTable.jsp, Names of inquiry administrative objects separated by commas. Only one inquiry is allowed for emxTableEdit.jsp. inquiry=SCSBuyerDesk,SCSBuyerDeskAssigned inquiry=ENCAllParts, ENCReleasedParts
inquiryLabel emxTable.jsp	This parameter is used with the "inquiry" parameter. Specifies the labels for each inquiry in the filter list in the table page header. Each label is associated to the corresponding inquiry administrative object name specified in the "inquiry" parameter. The label may have the actual text or the string resource id for internationalization.	One or more labels (equal to the number of objects specified in the "inquiry" parameter) for the inquiry administrative object names separated by a comma. inquiryLabel=All,Assigned inquiryLabel= emxEngineeringCentral.Common.All, emxEngineeringCentral.Common.Released
jsTreeID emxTree.jsp	The ID of the existing tree object, which is required to update the tree with "insert" mode. When the tree is constructed for the first time, the "jsTreeID" parameter will not have any valid value. Whenever a category within the tree is clicked, the system passes the current "jsTreeID" to that page and if that page needs to update the tree, it must call emxTree.jsp with the same "jsTreeID" value.	Active NodId, for example, root_7 or root_7_0_6 node567590204694.5947
jpoAppServerParamList emxTable.jsp emxForm.jsp emxIndentedTable.jsp	Allows session data to be passed to a JPO and uses the format: <code>scope:attributeName</code> where scope can be one of these values: <code>application</code> <code>session</code> <code>request</code> and the attribute must be a valid attribute used within the specified scope. The parameter can pass a comma-separated list of scope:attributeName values. The attribute values must be serializable.	application: <attributeName>, session: <attributeName>, request: <attributeName>
labelDirection	Used for bar charts, stacked bar charts, and line charts to specify whether to display the x-axis in a horizontal or vertical direction.	horizontal - Draws the x-axis label horizontally (default)

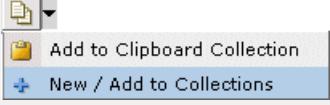
emxChart.jsp		vertical - Draws the x-axis label vertically
LASTREVISION	When true, the Highest check box for the Revision field is selected when the page opens. When false, the check box is clear.	true false
emxFullSearch.jsp	Default value is the value for the <code>emxFramework.FullTextSearch.LASTREVISION</code> property in <code>emxSystem.properties</code> . Passing this URL parameter overrides the default value set in <code>emxSystem.properties</code> .	
LATESTREVISION	When true, the By State check box for the Revision field is selected when the page opens. When false, the check box is clear.	true false
emxFullSearch.jsp	Default value is the value for the <code>emxFramework.FullTextSearch.LATESTREVISION</code> property in <code>emxSystem.properties</code> . Passing this URL parameter overrides the default value set in <code>emxSystem.properties</code> .	
launched		true false (default)
emxTable.jsp emxForm.jsp	When the Launch button is clicked from a channel tab, <code>launched=true</code> is passed to the new popup page. It indicates that the popup is a result of clicking the Launch button. The behavior of the popup page can be changed from the normal mode pages using this parameter. In normal mode, pages will not have the launched parameter and will default to <code>launched=false</code> .	
level	The number of hierarchical levels to traverse to fetch object details. The level must be an integer value.	1 (default)
emxStructureCompare.jsp		
lookupJPO	The JPO name and method name to execute the lookup function for a row added to the structure browser (using +). The JPO is invoked when the user clicks Lookup Entry .	JPOName:methodName
emxIndentedTable.jsp		
mandatorySearchParam	A comma-separated list that defines mandatory search criteria. The user will not be able to de-select any criteria specified.	Type Originator,Policy
emxFullSearch.jsp		
massPromoteDemote	Overrides the system property <code>emxFramework.Lifecycle.MassPromoteDemote.Enable</code> for the specific page. The default value for this parameter is the value for the above property. The page must also include the State and Type columns.	true false
emxTable.jsp emxIndentedTable.jsp	Use to turn on or off mass update controls on the editable table page.	true (default)--Mass update is available for the page. false--Mass update not is available for the page.
emxTable.jsp emxIndentedTable.jsp		
matchBasedOn	One or more column names from the table. The value can be a comma-separated list.	Type,Revision
emxStructureCompare.jsp		
MenuName	Defines which application submenu should be expanded when the Navigator page is called. If you don't specify a submenu, the first menu in the My Desk menu will be expanded.	The name of a menu object that represents an application submenu. For example: <code>emxNavigator.jsp?MenuName=TMCMYDesk</code>
emxNavigator.jsp	Can also be used along with the CommandName parameter to define the page to appear in the Content frame on login.	
mode	Specifies whether the form or structure browser page should be view or edit mode.	view (default)--form or structure browser is read only edit--form or structure browser is editable <code>emxForm.jsp?form=ENCPart&mode=edit</code>
emxForm.jsp emxIndentedTable.jsp		
mode	Controls whether the tree for the object is inserted into the current tree or replaces the current tree.	Insert--Insert the tree for the selected object into the current tree, if there is one. Replace (default) --Replace the current tree, if there is one, with the tree for the selected object.
emxTree.jsp		
mode	Used when calling the Navigator page from an external link on a Web site, portal page, or email message. Defines whether the Navigator window contains the banner and global toolbar as it normally does when using the applications.	Menu--The Navigator page contains all standard elements. Tree--The Navigator page contains only the tree frame and the content frame. For example: <code>emxNavigator.jsp?mode=Menu&ContentPage=../sourcingcentral/emxBuyerDeskTable.jsp</code>
emxNavigator.jsP		
multiColumnSort	Enables or disables multiple column sorting on a page. If disabled, the user can still	true (default)

emxTable.jsp emxIndentedTable.jsp	sort by a single column if one is specified in the sortColumnName parameter. When set to false, multiple column sorting is also disabled in any custom tables users' create based on this system table.	false
nameField	Configures the type of name field on the generic create form.	autoName keyin (default) both
emxCreate.jsp		
objectBased	Use to configure a table page so it lists items other than business objects. For example, the items such as collections and IconMail message are not business objects but can be listed in a table by passing the URL parameter objectBased=false. When <code>objectBased</code> is set to false, the parameter <code>objectCompare</code> is always assumed as false and the object compare icon is not displayed. Also, when <code>objectBased</code> is set to false, the JPO that gets the list should follow the same guidelines used for getting the object list. But the MapList containing the HashMaps with key name "id" should be assigned to the row identifier value and not the "objectId." The value assigned to "id" is used as the row identifier for every row in the table. If the table has a check box column (named emxTableRowId), the check box value is assigned with the value in "id".	true (default)--The table page lists business objects. false--The table page lists items that are not business objects.
emxTable.jsp		
objectCompare	Use to display or hide the object compare icon. This parameter is assumed as false and the object compare icon is not displayed if either of the following conditions apply: URL parameter <code>objectBased</code> is set to false. URL parameter <code>selection</code> is set to single or none.	true (default)--The object compare icon is displayed. false--The object compare icon is not displayed.
emxTable.jsp		
objectId	Use to specify the business object for which the properties need to be displayed. Required parameter for structure browser. Generic Create form: If an OID is passed, the object created will be connected to this object using the relationship specified by the relationship parameter.	<OID for the root object in the structure> 2233.5567.2323.4678 For example: emxForm.jsp?objectId=3243.32424.232
emxForm.jsp emxCreate.jsp emxIndentedTable.jsp emxStructureCompare.jsp		
objectId	When used with emxTree.jsp, defines the business object ID for which the tree will be constructed. For example, if the object is of type ECR, the tree object of menu type "type_ECR" will be constructed. If the menu "type_ECR" does not exist, the tree will be based on the "Default Tree" configuration. When used with emxNavigator.jsp, defines the business object whose navigation tree should be displayed when the Navigator window is called.	Valid business object ID. 2233.5567.2323.4678 emxForm.jsp?objectId=3243.32424.232
emxTree.jsp emxNavigator.jsp configurable pages, many standard pages, most custom pages		
ObserveHidden	Determines whether to display hidden types or not.	true (default)--Hidden types are not included in the chooser's list of types. false--Hidden types are included.
emxTypeChooser.jsp		
onReset	The name of a JavaScript function that is invoked when a user clicks the Reset button on an editable Structure Browser page. The function is invoked after resetting all changes on the page, but before refreshing the structure. The function must be included in the file defined in <code>emxSystem.properties: eServiceSuiteAPPLICATIONNAME.UIfreezePane.ValidationFile = VALIDATIONFILENAME</code> (either .js or .jsp) For example: <code>eServiceSuiteEngineeringCentral.UIfreezePane.ValidationFile = emxEngineeringCentralFormValidation.jsp</code>	resetEverything
emxIndentedTable.jsp		
owner	Assigns a default owner to the object being created. If no value is passed, the context user is assigned as the owner.	Design Engineer Buyer
emxCreate.jsp		
pageSize	An integer defining the initial page size (number of rows) for the search results. The parameter overrides the emxFramework.FullTextSearch.PageSize property.	50 500
emxFullSearch.jsp		
pagination	Specifies the number of rows to show per page. If the rows span more than one page, users can use the Left and Right Arrow buttons or the Page drop-down list to navigate to other pages.	Any number. Use 0 if you don't want to paginate. If 0 is specified, the pagination controls do not include the controls for navigation.
emxTable.jsp, emxFormEditDisplay.jsp	If this parameter is not specified, the system uses the value set in the pagination property in emxSystem.properties. When installed, this property is set to 10 rows per page. If 0 is specified, all objects are listed on one page.	

parentOID	Passed to toolbars if there is a valid objectId passed in to emxTable.jsp (in case of tree calling the table page).	Business object ID, for example, 46697.12656.52860.3882
configurable pages, many standard pages, most custom pages		
policy	The policy to assign to the type being created. You can use either the original or symbolic policy name, although symbolic names are recommended.	policy_Part policy_DifferentPart
emxCreate.jsp		
portal	Used when calling the Navigator page from an external link to ensure the Login page does not replace the frame that contains the link.	true--Turns the portal mode on to ensure the Login page and Navigator page have no effect on the page that contains the URL link to the Navigator page. false--Turns the portal mode off. If the parameter isn't specified, the portal mode is turned off. .//Apps/Framework/VERSION/commonUI/emxNavigator.jsp?portal=true
emxNavigator.jsp		
portal	Specifies the menu administrative object that represents the portal page.	Name of menu administrative object defined to represent a portal page.
emxPortal.jsp		
portalMode	Every page configured inside the PowerView includes the parameter <code>portalMode=true</code> , so that the page can differentiate between normal display and portal display.	true false (default)
emsForm.jsp	When the Launch button is clicked, it launches the currently displayed channel tab into a maximized popup window, passing the parameters <code>launched=true</code> and <code>portalMode=false</code> to the new window.	
portaltable	Specifies the alternate table that can be used within a channel tab.	Name of alternate table
emxTable.jsp	Displaying table component listings in half page-width channels will typically cause the user to scroll horizontally to the view all columns in the table. For this situation, you can create an alternate table with fewer columns for display specifically within a channel tab. For example, if a standard ECO New Parts table contains 10 columns, you could define an alternate ECO New Parts table that contains just 4 columns.	
postProcessJPO	The JPO to use when a user clicks the Done button on a table or form component.	<JPOName>:<MethodName>
emxFormEdit.jsp emxTableEdit.jsp emxIndentedTable.jsp		
postProcessURL	The JSP called when a user clicks the Done button on a table or form page. The configured JSP is called after the edit processing and database update. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.	`\${SUITE_DIR}/emxCustomPostProcess.jsp or ..</ApplicationDirectory>/emxCustomPostProcess.jsp Example: .../engineeringcentral/emxCustomPostProcess.jsp
preFilter	Determines which action types to display when the History page comes up. If the parameter is not passed, the page lists all action types. If the Action Type filter control is shown on the page (determined by the ShowFilterAction parameter), the preFilter actions are listed in the text box and the filter is applied to the history list when the page first opens. The user can use the Action Type filter to change the filter, adding and removing displayed actions as needed. If the Action type filter control is not shown, the user cannot change the preFilter actions. By setting preFilter actions and hiding the Action Type filter, you can hide specific actions that you do not want users to see.	The parameter accepts two types of values: Comma delimited list of action types: <code>preFilter=connect, disconnect, create</code> String Resource ID: <code>preFilter=emxSystem.preFilter.List</code> To specify which string resource properties file to look in to get the value, include the SuiteKey parameter. Possible values for the SuiteKey parameter are: eServiceSuiteTeamCentral, eServiceSuiteEngineeringCentral, eServiceSuiteProgramCentral, or any other suite name. If the SuiteKey is not passed in, the History page looks for the key in the emxSystem.properties file. For example, in the emxSystem.properties file, the value for the emxSystem.preFilter.List might be "connect, create, disconnect".
emxHistory.jsp		
preProcessJPO	The JPO to use when a user clicks Edit on a table or form component.	<JPOName>:<MethodName>
emxFormEdit.jsp emxTableEdit.jsp emxIndentedTable.jsp		
preProcessURL	The JSP called before displaying an editable table or form. Specify the JSP name using the macro for the page location, or use the relative path as listed in the examples.	`\${SUITE_DIR}/emxCustomPreProcess.jsp or ..</ApplicationDirectory>/emxCustomPreProcess.jsp Example: .../engineeringcentral/emxCustomPreProcess.jsp
emxFormEdit.jsp emxTableEdit.jsp emxIndentedTable.jsp		

PrinterFriendly	Specifies whether the page should include the Printer Friendly tool. If this parameter is not included, the value is assumed to be True and the page displays the Printer Friendly tool.	true (default) false
emxTable.jsp emxForm.jsp emxIndentedTable.jsp configurable pages, many standard pages, most custom pages	You can have the system pass this parameter automatically by entering the PrinterFriendly setting for the command object that calls the page.	
program emxTable.jsp emxChart.jsp, emxFormEditDisplay.jsp	<p>Used only when there is no <code>inquiry</code> parameter passed in to emxTable.jsp or emxTableEdit.jsp.</p> <p>This parameter is used as an alternative approach to the inquiry object approach. The <code>program</code> approach uses a JPO program object to fetch the object list to be displayed in the table.</p> <p>For emxTable.jsp, the parameter value can have one or more sets of values separated by a comma ",".</p> <p>For emxTableEdit.jsp, multiple program parameters are not supported. If multiple programs are passed with the ',' separator, only the first one will be used and rest are ignored.</p> <p>emxTable.jsp uses the first set of values (JPO name and method name) to display the list in the table when the page is first loaded. It uses all the sets for the table filter list in the header.</p> <p>If there is only one program, the filter in the header is not displayed.</p> <p>Currently, to build the filter list, you must use either JPOs or Inquiry objects. You cannot use both.</p> <p>Charts: Used only when there is no <code>inquiry</code> parameter passed to emxChart.jsp.</p> <p>This parameter is used as an alternative approach to the <code>inquiry</code> object approach. This <code>program</code> approach uses the JPO program object to get the object list to be displayed in the table.</p> <p>This parameter is assigned to one set of values that will form a JPO program name and the method name. The format of the parameter value is:</p> <pre>program=<JPO program name>:<JPO method name></pre> <p>The program name and the method name are separated by a colon ":".</p> <p>The parameter value may have one or more sets of values separated by comma ",".</p> <p>The first set of values (JPO name and method name) are used to get the object list when loading first. All programs are associated with the table filter in the header.</p> <p>If there is only one program value, the filter in the header will not be shown.</p>	<p>This parameter is assigned to one (or more in view mode) set of values that will form a JPO program name and the method name. The format of the parameter value is: <code>program=<JPO program name>:<JPO method name></code></p> <p>The program name and the method name are separated by a colon ":".</p> <pre>program=emxTableBuyerDesk:getBuyerDesk,emxTableBuyerDesk:getAssignedBuyerDesk</pre>
programLabel emxTable.jsp	<p>Used with the parameter "program" defined above. Specifies the labels for each program in the filter list in the table page header.</p> <p>The value may contain one or more labels (equal to number of "program" values) separated by a comma ",".</p> <p>Each label is associated to the corresponding JPO object and method name specified in "program" parameter.</p> <p>The label may have the actual text or the string resource id for internationalization.</p>	<pre>programLabel =All,Assigned</pre> <pre>programLabel = emxEngineeringCentral.Common.All, emxEngineeringCentral.Common.Released</pre>
queryLimit emxFullSearch.jsp	For the specific full-text search page, overrides the limit defined by the <code>emxFramework.FullTextSearch.QueryLimit</code> property in <code>emxSystem.properties</code> for the maximum number of results returned from a search	<integer value>
queryType emxFullSearch.jsp	Default value is defined by the <code>emxFramework.FullTextSearch.QueryType</code> property in <code>emxSystem.properties</code> .	Real Time Indexed
RegisteredDirectory emxForm.jsp	The directory where the help files are located.	Directory name within ematrix.
relationship emxIndentedTable.jsp emxCreate.jsp emxStructureCompare.jsp	<p>Structure Browser: Relationship names to be used for expanding the root object to display the table structure view. The same relationships are used while expanding the structure by clicking on the plus on the nodes.</p> <p>The relationship name passed in can be the symbolic name or the actual relationship name. Using the symbolic names is recommended.</p> <p>A property setting can be used to pass the relationship names. The property must be assigned to one or more relationships to be used for expansion.</p>	<pre><relationship name> <list of comma separated relationship names> <property key assigned to one or more relationships></pre> <p>For example:</p> <pre>relationship_EBOM, relationship_PartSpecification</pre> <pre>relationship_Employee</pre> <pre>all (default)</pre>

	<p>When one or more relationships is passed in, the program uses the list of passed-in relationship(s) only, to expand the object. If no relationship parameter is passed in, then it assumes all.</p> <p><all> (default) - The given object will be expanded to get all the connected objects, irrespective of what relationship is used, to be displayed as child/parent objects.</p> <p>Generic Create Form: The name of the relationship to use to connect the object being created to the object passed by the objectId parameter. You can use either the original or symbolic relationship name, although symbolic names are recommended.</p>	For Generic Create form, a single relationship can be passed.
relationshipFilter	Used to turn ON or OFF the relationship filter shown in the header. By default the parameter is false and the relationship filter is hidden. The parameter must be passed explicitly as true to show the filter.	true false (default)
emxIndentedTable.jsp		
relID	Use to specify the relationship used to get the relationship attribute values for relationship fields.	Valid relationship ID. emxForm.jsp?relId=3243.32424.232
emxForm.jsp		
ReloadOpener	Determines whether to call a reload() method on the opener/parent page (the form page from which the choose is launched). Reloading the page updates other fields on the form page based on the type(s) selected from the Type Chooser.	true--The reload method is called. false (default)--The reload method is not called.
emxTypeChooser.jsp		
rememberSelection	Controls whether the system remembers the items a user selects on one page when the user navigates to another page in the table. The emxSystem.properties key emxNavigator.UITable.Pagination.RememberSelection sets the default for table pages. Use this parameter to override the default for a single page.	true--Selected items are remembered across the table's pages. false--Selected items are not remembered across pages.
emxTable.jsp		
renderPDF	Controls whether the Render PDF icon is shown in the form toolbar when displaying the form in View mode.	true--The Render PDF icon is shown in the form toolbar.
emxForm.jsp	If the parameter is not passed in, the default is false and the icon is not shown.	false--The Render PDF icon is not shown in the form toolbar.
reportType	The type of report to generate.	complete_summary (default) difference_only Unique_toLeft_Report Unique_toRight_Report Common_Report
emxStructureCompare.jsp		
resequenceRelationship	Used when the order of child objects is significant. Defines a comma-separated list of relationships that determine if a paste operation is a resequencing. Resequencing means that the child object was moved from one place in the list of child objects to another (under the same parent object).	relationship_AffectedItems, relationship_EBOM
emxIndentedTable.jsp		
searchCollectionEnabled	When true, includes the Search in Collection toolbar button for navigation-based searches.	true (default) false
emxFullSearch.jsp	This parameter is not recommended when searching type-specific tables, or results tables that use expandJPOs or relationships.	
SelectAbstractTypes	Determines whether users can select abstract types.	true--Abstract types can be selected. false (default)--Abstract types cannot be selected but they are displayed in the hierarchical list of types.
emxTypeChooser.jsp		
selection	Controls whether the table page adds a column of check boxes or radio buttons in the left-most column of the table.	multiple--Users can select more than one row in the table. A check box is displayed in the left column of each row. There is no access restriction for the check boxes. If this parameter is passed, it overrides any check box column added to the table administrative object.
emxTable.jsp	When this parameter is set to <code>single</code> or <code>none</code> , the parameter <code>objectCompare</code> is always assumed as false and the object compare icon is not displayed.	single--Users can select one row in the table. A radio button is displayed in the left column of each row.
emxIndentedTable.jsp		
emxFormEditDisplay.jsp		
emxFullSearch.jsp		none--A selection column is not added to the table page. (A check box column can still be displayed by adding the column to the table administrative object.)
SelectType	Determines whether the Type Chooser has single select radio buttons or multi-select check boxes.	multiselect--Users can choose multiple types using check boxes. singleselect--Users can choose only one type using a radio button.
emxTypeChooser.jsp		
any search JSP page	singleselect is the default for the Type Chooser; multiselect is the default for search pages.	
showApply	When set to true, shows the Apply button in Edit mode. When set to false, Edit mode does not have an Apply button.	true (default) false
emxIndentedTable.jsp		
showClipboard	This menu is enabled by default and adds the  to the page toolbar. The icon acts	true (default)

emxIndentedTable.jsp emxTable.jsp emxTableEdit.jsp emxSearchConsolidatedView.jsp emxForm.jsp	<p>as a pull-down menu:</p>  <p>If the user clicks , selected objects are added to the clipboard collection for that user. If the user clicks the arrow, the user can select the New/Add to Collections or Add to Clipboard Collection command.</p> <p>Set the value for this parameter to false to disable this feature.</p>	false
ShowFilterAction	Determines whether to display the action type filter or not.	true (default)--The action type filter displays. false--The action type filter does not display.
emxHistory.jsp		
ShowFilterTextBox	Determines whether to display the filter text box or not.	true (default)--The filter text box displays. false--The filter text box does not display.
emxHistory.jsp		
ShowIcons	Determines whether to display types icons in the Type Chooser.	true (default)--Type icons are displayed. false--Type icons are not displayed. This improves performance.
emxTypeChooser.jsp		
showInitialResults	If true, shows an unfiltered result set when the full-text search (navigation mode) is opened. If false, the user must select initial criteria and click Search and then a result set is displayed.	true
emxFullSearch.jsp	This value overrides the system property emxFramework.FullTextSearch.ShowInitialResults set in emxSystem.properties.	false
showPageHeader	Enables the page header in the PowerView page. When set to false, the header(including the toolbar) does not display in the PowerView page.	true (default) false
emxPortal.jsp		
showPageURLIcon	When true, shows in the toolbar. This tool lets users copy the URL to the specific ENOVIA application page. When false, ICON does not show in the toolbar.	true
emxFormEditDisplay.jsp emxFullSearch.jsp emxForm.jsp emxCreate.js emxTable.jsp emxTableEdit.jsp emxIndentedTable.jsp emxPortal.jsp	The default value for this parameter is defined by the <code>emxFramework.Toolbar.ShowPageURLIcon</code> property in <code>emxSystem.properties</code> .	false
showRMB	Enables or disables the right-click menus on the page. When set to false, all right-click menus for that page are disabled.	true (default) false
emxTable.jsp emxIndentedTable.jsp		
showSavedQuery	Includes the AEFSaveQuery command on the page toolbar.	true (default) false
emxFullSearch.jsp		
showTabHeader	Applies only in Portal mode (when the page is displayed within a PowerView), enables or disables the page header.	true
emxIndentedTable emxTable.jsp .jsp emxTableEdit.jsp emxForm.jsp	If false, the header text is not displayed in the PowerView tab. If true, the header text shows in the tab.	false (default)
sortColumnName	Specifies the column by which the table should be sorted when the page is first loaded. If no column is specified, the rows are listed in the order they are retrieved from the database.	Name of table column. column1,column2,column3
emxTable.jsp emxIndentedTable.jsp, emxFormEditDisplay.jsp	Can be a comma-separated list of up to 3 column names. The page sorts by the first provided column, then by the second, then by the third. Used as the default Sort by settings in the Customize Table View dialog box. Columns defined with sortType=other setting cannot be used with multiColumnSort.	

sortDirection	Defines the sort order for the columns specified in the sortColumnName parameter. If a single value is passed, it applies to all columns, or you can pass a comma-separated list of values matching the number of columns in the sortColumnName parameter. Used as the default sort directions in the Customize Table View dialog box.	ascending (default)--Sort a to z or 0 to n. descending--Sort z to a or n to 0. ascending,descending,ascending
StringResourceId	The name of the String Resource file for the application.	emxFrameworkStringResource
configurable pages, many standard pages, most custom pages		
Style	Determines the cascading style sheet to use for defining the layout and color for headers and footers on the table page or popup dialog. If the style parameter is not specified, the <i>list</i> style is used with the Configuring Styles style sheet. You can use a different style sheet by changing the following property in emxSystem.properties: <code>emxNavigator.UITableView.Style.List = styles/emxUIList.css</code> If you specify <i>style = dialog</i> , the Configuring Styles style sheet is used. You can use a different style sheet by changing the following property in emxSystem.properties: <code>emxNavigator.UITableView.Style.Dialog = styles/emxUISearch.css</code> If you specify <i>style = PrinterFriendly</i> , the Configuring Styles sheet is used. This is hard-coded into the emxTableReportView.jsp page, which is called to display the printer-friendly page.	list (default)--For table pages displayed in the content frame of emxNavigator.jsp. dialog--For table pages displayed in popup windows, such as search result pages. PrinterFriendly--This style is used when displaying a page in printer-friendly mode directly from a command link, rather than from the Printer Friendly icon in a list page.
subHeader	Only needed when using All Revisions mode. Determines whether to display Version or Revision for the SubHeader, which separates the history events for each revision.	Version--The History page looks in the emxFrameworkStringResource.properties file for value of the emxFramework.History.Version property. Revision--The History page looks in the emxFrameworkStringResource.properties file for value of the emxFramework.History.Revision property.
subHeader	Creates a subHeader below the main header in the table header frame.	The value can be any static text or a string resource id. For example: subHeader= emxEngineeringCentral.Common.BOMLevel subHeader=Bill of Material Level 1 The value can also include macros such as \$<type> \$<revision>.
submitAction	Determines the action to take after creating the object (in addition to closing the form window): refreshCaller: Reload the calling page. If called from a table or structure browser, then reloads the table or structure browser. doNothing: When called from a table or structure browser, does not refresh (including sorting) the calling page. treeContent: Load the newly-created object's tree in the main content frame. treePopup: Load the newly-created object's tree in a new pop-up window. For emxCreate.jsp, if no value is passed for this parameter, this alert displays to the user: <code>The object Type <type name> Name:<object name> Rev:<revision> is created successfully.</code>	refreshCaller treeContent treePopup
SubmitLabel	Defines the label for the Submit link. If not passed, the label shows as "Done". You can pass a static text string or a string resource value (recommended).	Any static text or string resource id. SubmitLabel=emxFramework.GlobalSearch.Select=Select
SubmitURL	Displays the Submit link in the table footer frame. The value can be any valid JSP page, which gets run upon clicking Submit.	Any JSP. The path can include any directory macro. SubmitURL=\${SUITE_DIR}/emxBlank.jsp
emxTable.jsp, emxIndentedTable.jsp, emxFormEditDisplay.jsp emxFullSearch.jsp	The label for the Submit link can be configured using the SubmitLabel parameter. If SubmitLabel is not passed in, the default label "Submit" is used.	
SubmitURL	Holds the URL that the result page should call when the submit button is pressed.	AnyPage.jsp
emxFullSearch.jsp any search JSP page		

SuiteKey	Determines which string resource property file to look in for the preFilter property key. This parameter is only used when the preFilter parameter is passed in as a resource ID. For example, SuiteKey=eServiceSuiteEngineeringCentral would be required for the preFilter parameter whose value is in the EngineeringCentral properties file, emxEngineeringCentral.properties.	eServiceSuiteTeamCentral, eServiceSuiteEngineeringCentral, eServiceSuiteProgramCentral, or any other suite name.
emxHistory.jsp emxTypeChooser.jsp configurable pages, many standard pages, most custom pages	For the Type Chooser: Determines which application's properties file to look in for the key specified in the InclusionList or ExclusionList parameter. If the SuiteKey is not passed in, emxSystem.properties is used.	
table	Specifies the table object to use for presenting this targeted page or the structured view. The table object has all the information needed to display the table, including the columns to present. For the full-text search, the AEFGeneralSearchResults table is used by default.	Name of table administrative object. For example: table=SCSBuyerDesk table=ENCParts table=AEGGeneralSearchResults
tableMenu	Can be assigned to an administrative menu or command name, which will contain the table name definition and label.	Administrative menu name Administrative command name
emxIndentedTable.jsp	When command is passed in: Command setting <code>table</code> is used to get the table name for the table definition. The Table Filter is not shown as there is only one table available to view. When menu is passed in: Commands connected to this menu are used for listing the options in the Table Filter combo box. The label for each item is obtained from the command's label. The setting <code>table</code> on individual commands is used as the table definition for displaying the view, upon selecting the options from the Table Filter. By default, the first command is used for the table definition. The commands connected to the menu honor all the access settings supported by the configurable component (such as Access Mask, Access Expression, and Access Program).	For example: ENCBOMViews (menu) PMCFolderViews(menu) ENCBOMView (command)
TipPage	Specifies whether the page should include the Tip Page tool the specific html or jsp to call when a user clicks the tool. If this setting is not included, the Tip tool is not included on the page.	Name of a custom html or JSP page, including any path. The starting point for the directory reference is the content directory. For example, if you want to call an html file in ematrix/doc/customcentral and the content directory is ematrix/customcentral, you would add this parameter to the jsp: TipPage=../doc/customcentral/tippage.html emxForm.jsp?form=ENCPart&TipPage=../myapplication/showMyTipPage.jsp?
emxForm.jsp emxCreate.jsp emxTable.jsp emxIndentedTable.jsp emxPortal.jsp configurable pages, many standard pages, most custom pages	Specifies the menu administrative object that represents the Actions menu, which appears in the page header. Defines the name of the toolbar menu to be shown on the table page. This can be passed to either emxTable.jsp or emxTableEdit.jsp. To use the same toolbar in both a view mode table and an editable, pass the same toolbar name to each. If the Edit button in the Table component is shown by passing the parameter editLink = true to emxTable.jsp, then the editToolbar parameter should be passed to emxTable.jsp. If the Edit button in the Table component is a custom command where the URL is a custom URL (for example emxTableEdit.jsp), then the parameter should be appended to the custom URL (emxTableEdit.jsp). For a form, specifies the menu administrative object used in the header frame. For a structure browser, a comma-separated list of UI menu names that add custom filters to the toolbar. Menus specified here are added as a new toolbar row beneath the Action toolbar.	Name of menu administrative object defined to represent the Actions menu. toolbar=SCSBuyerDesktopToolbar toolbar=ECEBOMToolbar,ECEBOMFilter
TransactionType	Controls whether the table query is run within an Update transaction or Read transaction. Update transaction can be used whenever the code needs to update the database while fetching the object list.	read (default)--The table query is not run within an Update transaction. update--The table query (inquiry or JPO) will be run within an Update transaction.
emxTable.jsp emxIndentedTable.jsp		

treeLabel	Use to override the default label for a tree, which is defined in the "Label" parameter of the tree menu administrative object. When this parameter is available to the emxTree.jsp page, the page uses this value as the Label for tree menu base node. This parameter also overrides any JPO specified for the label in the settings Label Program and Label Function.	emxTree.jsp?treeLabel=custom Part Label&objectId=3434.345.4564.7755
treeMenu	Use to override the standard tree menu for the object's type (or any alternate tree defined by an application). The standard tree menu for an object is named using the symbolic name of the type. For example, the standard tree menu object for quality part plans is type_QualityPartPlan. Using the custom tree menu by overriding the system-defined tree for any object type is NOT RECOMMENDED.	Name of a menu object for a tree: customPartTree
triggerValidation	Controls the Validate command on the toolbar. If true or empty, the command shows on the toolbar. If set to false, the trigger validation icon does not display on the toolbar.	true (default) false
emxIndentedTable.jsp		
txtExcludeOIDs	Comma-separated list of Object IDs (OIDs) to exclude from the search results.	
emxFullSearch.jsp		
txtTextSearch	If a value is passed using this parameter, that value is used as a search criteria for the initial load of the search page. The search criteria/results page loads with the results of this initial search, and the user can continue entering criteria as needed.	string
emxFullSearch.jsp		
type	Structure Browser: Used to pass the type name for filtering the expanded structure when the table is displayed and also when the structure is expanded. The type name passed in can be the symbolic name or the actual type name. Using the symbolic name is recommended. A property setting can be used to pass the type names. The property must be assigned to one or more types to be used for filtering. When one or more types is passed in, the program uses the list of passed-in types to filter the object list. If there is no parameter passed in, then it assumes all. all (default) - This assumes that all the objects expanded will be displayed and no filtering happens. This is an optional parameter. Generic Create Form: A comma-separated list of types that can be created by this form. The type can be the original or symbolic name of the business object, although symbolic names are recommended.	<type name> <list of comma separated type names> <property key assigned to one or more types> For example: type_Part,type_ECO type_Folder all (default)
emxIndentedTable.jsp		
emxCreate.jsp (required)		
typeChooser	Specifies whether or not the type chooser will be used for the Type field. If true, the default type chooser, emxTypeChooser.jsp, is used. Only the types defined by the type parameter will be selectable by the type chooser.	true false (default)
emxCreate.jsp		
typeFilter	Used to turn ON or OFF the type filter shown in the header. By default the parameter is false and the type filter is hidden. The parameter must be passed explicitly as true to show the filter.	true false (default)
emxIndentedTable.jsp		
useTypeChooser	setting that tells the search component whether to display a type chooser or not for the type field. If this parameter is not passed into the search component, then the default will be true.	true false
any search JSP page		
vault	The vault where the object being created will be stored. You can use either the original or symbolic value name, although symbolic names are recommended.	Gold vault_Gold
emxCreate.jsp		
vaultChooser	Specifies whether or not the vault chooser will be used for the Vault field. If true, the default type chooser, emxVaultChooser.jsp, is used.	true false (default)
emxCreate.jsp		
viewFormBased	Overrides the emxFramework.FullTextSearch.FormBased property defined in emxSystem.properties. True uses the form-based search; false uses the navigation-based search. If you set this value to true, you must also pass the formName parameter.	true false (default)
emxFullSearch.jsp		
XAxis	Table column name to be used as the x-axis for bar charts or stacked bar charts	Count
emxChart.jsp	Table column name to be used as the "slice" label for pie charts Table column name to be used as the line label for line charts This should be a non-numeric column except for line charts.	

YAxis	Comma-separated list of numeric column names to be used as the y-axis for bar charts, stacked bar charts, or line charts	totalCount
emxChart.jsp	Table column name to be used as the pie data for pie charts	weight

Parameters Automatically Passed to URLs

When a user chooses a command--such as a menu option or tree category--the system automatically appends one or more parameters to the URL specified in the href parameter for the object.

The target URL can use these parameters if needed or not. The configurable pages and many standard pages use one or more of these parameters and most custom pages will also. For information about which parameters are used by a JSP, refer to the section that describes the JSP.

Parameter	Description	Values Sent
CurrencyConverter	<p>Passes the value entered for the Currency Converter setting. If True, the target page should include the Currency Converter tool. If the setting is not included, the value is assumed to be false and the target page does not display the Currency Converter tool.</p> <p>This setting should only be used with Sourcing Central.</p>	Value specified in Currency Converter setting: True False
emxSuiteDirectory	The name of the directory for the application under the ematrix directory.	engineeringcentral
HelpMarker	Passes the help marker text specified for the Help Marker setting.	Value specified in Help Marker which is a help marker for the page.
jsTreeID	<p>The ID of the existing tree object, which is required to update the tree with "insert" mode.</p> <p>When the tree is constructed for the first time, the "jsTreeID" parameter will not have any valid value.</p> <p>Whenever a category within the tree is clicked, the system passes the current "jsTreeID" to that page and if that page needs to update the tree, it must call emxTree.jsp with the same "jsTreeID" value.</p>	Active NodId, for example, root_7 or root_7_0_6
objectId	<p>The business object ID for the selected object.</p> <p>JSPs use the objectId to display information for that object. For example, emxTree.jsp uses it to construct the tree for that object type. If the object is of type "ECR", the tree object of menu type "type_ECR" will be constructed.</p>	Business object ID, for example, 46697.12656.52860.3882
parentOID	Passed to toolbars if there is a valid objectId passed in to emxTable.jsp (in case of tree calling the table page).	Business object ID, for example, 46697.12656.52860.3882
PrinterFriendly	<p>Passes the value specified for the Printer Friendly setting. JSPs that use the PrinterFriendly parameter, such as emxTable.jsp and emxForm.jsp, show the Printer Friendly tool when the setting is True and hide it when False. If the setting is not included, emxTable.jsp and emxForm.jsp show the tool by default. Users can click the tool to get a version of the current page that can be printed with the browser's Print button.</p>	Value specified in Printer Friendly setting: True False
relId	<p>The relationship ID for the selected relationship.</p> <p>JSPs use the relId to display information for that relationship.</p>	
StringResourceFileId	The name of the String Resource file for the application.	emxFrameworkStringResource
suiteKey	Passes the property name from emxSystem.properties that maps to the Registered Suite setting's value. For example, if the Registered Suite is EngineeringCentral, then the suiteKey value is ServiceSuiteEngineeringCentral.	Name of property in emxSystem.properties that maps to Registered Suite setting value.
TipPage	Passes the value specified for the Tip Page setting. If the setting includes a URL, the target page includes the Tip Page tool and the tool calls the URL when a user clicks it. If the setting is not included, the target page does not display the Tip Page tool.	Value specified in Tip Page setting which is the name of a custom URL page.

Selectables

You can use select expressions to obtain or use information related to a business object. This section defines the available selectables for ENOVIA objects.

In this section:

- [About Selectables](#)
- [Definition Objects](#)
- [Relationship Direction](#)
- [Selectable Fields for Applications](#)
- [Selectable Fields for Attribute](#)
- [Selectable Fields for Business Objects](#)
- [Selectable Fields for Commands](#)
- [Selectable Fields for Connections](#)
- [Selectable Fields for Dataobjects](#)
- [Selectable Fields for Dimensions](#)
- [Selectable Fields for Expressions](#)
- [Selectable Fields for Form Fields](#)
- [Selectable Fields for Formats](#)
- [Selectable Fields for Forms](#)
- [Selectable Fields for Groups](#)
- [Selectable Fields for Inquiry](#)
- [Selectable Fields for Interface](#)
- [Selectable Fields for Location](#)
- [Selectable Fields for Menus](#)
- [Selectable Fields for Person](#)
- [Selectable Fields for Policy](#)
- [Selectable Fields for Process](#)
- [Selectable Fields for Product](#)
- [Selectable Fields for Program](#)
- [Selectable Fields for Query](#)
- [Selectable Fields for Relationships](#)
- [Selectable Fields for Role](#)
- [Selectable Fields for Server](#)
- [Selectable Fields for Store](#)
- [Selectable Fields for Table](#)
- [Selectable Fields for Table Columns](#)
- [Selectable Fields for Type](#)
- [Selectable Fields for User](#)
- [Selectable Fields for Vault](#)
- [Selectable Fields for Webreports](#)
- [Selectable Fields for Wizard](#)
- [Selectable Fields for Workflow](#)

About Selectables

The following topics are discussed:

- [Select Expressions](#)
- [Scenario for Selectables](#)
- [Selectable Fields](#)

Select Expressions

You can use select expressions in several operations within the Studio Modeling Platform:

- In Matrix Navigator, you can use select expressions in queries, tables, and visuals.
- In Business Modeler, you can use select expressions in reports and forms.
- In MQL, you can use select expressions in print and expand statements, as well as queries.

The purpose of select expressions is to obtain or use information related to a business object. The system attempts to produce output for each select clause input, even if the object does not have a value for it. If this is the case, an empty field is output.

- In a query or a visual, the select expression value is used to qualify the search criteria (in a where clause) by comparing it with another (given) value.
- In a table, report, or form, the expression defines the information to present in a column or field.
- In a `print` or `expand` statement, the expression defines the information to output about the object(s).

You can obtain information which includes not only attribute values and other business object data, but also administrative object information, such as the governing policy, vault, and so on. The key property of a select expression is that it can access information related to an object.

In all cases, the expression is processed from the context of a starting object.

- In a query, the starting point is the business objects that meet other selection criteria (vault, type, and so on).
- In a table, the starting point is the business object in each row.

The phrase starting point is used because the select mechanism actually uses the same concept of navigation from one object to another that makes the rest of the system so flexible. Most information is actually represented internally by a small object and not by a text string or numeric value as it appears to the user.

These internal objects are all linked in the same way business objects are connected by relationships. The links can be navigated from one object to another. A period (.) indicates a link in the select expression. The entire list of selectable fields can be obtained via MQL.

If you type this MQL command:

```
print businessobject selectable;
```

The result is similar to:

```
name
description
revision
originated
modified
lattice.*
owner.*
grantor.*
grantee.*
granteeaccess
granteesignature
policy.*
type.*
attribute[].*
default.*
format[].*
```

```
current.*  
state[].*  
revisions[].*  
previous.*  
next.*  
first.*  
last.*  
history  
relationship[].*  
to[].*  
from[].*  
exists  
islockingenforced  
vault.*  
locked  
locker.*  
reserved  
reservedby  
reservedstart  
reservedcomment  
id  
method  
search*  
workflow[].*
```

The `.*` notation indicates the item is an object and you can navigate to other information from that object. Items without this notation are simple data types (`locked`, `id`, `method`, and so on) and cannot be used for further navigation.

The square bracket notation (for example, `attribute[].*`) indicates that there can be many linked attribute objects and that a specific name can be entered if it is known. By contrast, object notations without the square brackets (for example, `owner.*`), means there is only one Owner object linked to the starting object.

If you expand the information available from the Owner object by typing:

```
print businessobject selectable owner;
```

these options are listed:

```
owner.name  
owner.description  
owner.property[].*  
owner.hidden  
owner.id  
owner.mask.*  
owner.admin.*  
owner.email  
owner.isaperson  
owner.isagroup  
owner.isarole
```

`owner.name` indicates that you can find the name of the owner by navigating to the Owner object and then to the name data (string, in this case) in that object. Each successive expansion may reveal other objects that can yield more information.

Refer to the "Working with Workspace Objects" chapter of the *MQL Guide* for more information on second level selectables that may help improve expand and query performance.

A business object can only have a single governing policy, so the square bracket notation does not appear. If you expand the information available from the `policy.*` option by typing:

```
print businessobject selectable policy;
```

These options are listed:

```
policy.name  
policy.description  
policy.property[ ].*  
policy.hidden  
policy.id  
policy.type[ ].*  
policy.format[ ].*  
policy.defaultformat.*  
policy.state[ ].*  
policy.revision  
policy.store.*  
policy.islockingenforced
```

If you expand the information further for the `policy.state` option by typing:

```
print businessobject selectable policy.state;
```

These options are listed:

```
policy.state.name  
policy.state.publicaccess.*  
policy.state.owneraccess.*  
policy.state.notify  
policy.state.route  
policy.state.action  
policy.state.check  
policy.state.access[ ].*  
policy.state.signature[ ].*  
policy.state.revisionable  
policy.state.versionable  
policy.state.autopromote  
policy.state.checkouthistory
```

You use the same process for expanding selectable fields as the process of expanding connected business objects in the Navigator browser. It involves following links from one item to another.

Programs should be explicit about the selectables desired and their output order. ENOVIA does not guarantee output will remain in the same order from version to version.



Scenario for Selectables

This example shows a business object, Assembly 90000 A, with the following properties:

```
Assembly 900000 A  
policy = Production  
current = Released  
description = Bed Frame  
cost = 1000  
weight = 100
```

Example

Suppose you want to use the information about the policy. The table below shows how this information is obtained with the MQL print command and also how it could be used in finds, tables, and so on.

	MQL print Command	Find or Visuals Where Clause	Table, Report, Form Field
--	-------------------	------------------------------	---------------------------

Select Expression	<pre>print bus Assembly 90000 A select policy;</pre>	policy = production	policy
Result	MQL returns: <pre>policy = Production</pre>	Assembly 90000 A is found or the visuals are applied to it.	If Assembly 90000 A is in the table/report/form, Production is output in that column/field.

In the where clause, a space must be included before and after all operators.

Example:

This example shows how to use the attribute "Cost" in Assembly 90000 A. The table below shows how this information is obtained with the MQL print command and also how it could be used in finds, tables, and so on.

	MQL print Command	Find or Visuals Where Clause	Table, Report, Form Field
Select Expression	<pre>print bus Assembly 90000 A select attribute[cost];</pre>	<pre>type == Assembly && attribute[cost] == 1000 (or perhaps, attribute[cost] < 2500, and so on.)</pre>	attribute[cost]
Result	MQL returns: <pre>attribute[cost] = 1000</pre>	Assembly 90000 A is found or the visuals are applied.	If Assembly 90000 A is in the table, report, or form, 1000 is output in that column/field.



Selectable Fields

This section contains alphabetical listings of all the selectable fields currently supported. Each listing includes the data type, a description of its meaning, and the default output. Data types included are:

- Simple data types (string, integer, real, timestamp, boolean)
- Administrative objects (business definitions--policy, attribute, etc.)
- Internal objects (objects that are not seen by the user)
- Business objects
- Relationship instances

Each select expression has a default behavior to minimize the amount of typing required for an answer. In the case of simple data types, the default is always to return the value. Objects have different behavior in each case.

Definition Objects

Many of the selectable fields are definition objects (type, policy, vault, etc.). You access the definition information through the `select` mechanism.

This information is less likely to be used by ENOVIA product users, but is useful for system integrators who may have to build programs and need to understand the internal details of the schema being used. Since all definitions have some relationship to business objects, they can be accessed from them (and, therefore, from the application rather than the Business Modeler) when needed. For example, expanding:

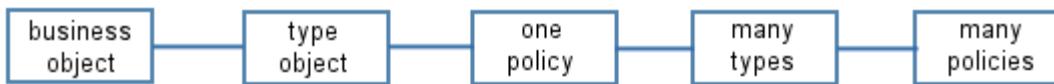
```
policy.type[].*
```

yields (among other things):

```
policy.type.policy[].*
```

A business object is governed by a policy. That policy can govern other object types. These types can have other governing policies.

The first `policy` is not followed by square brackets. This is because a business object instance can have only one governing policy. By contrast, the second `policy` is followed by square brackets because a type definition can have many optional policies. The connections can be illustrated as:



Relationship Direction

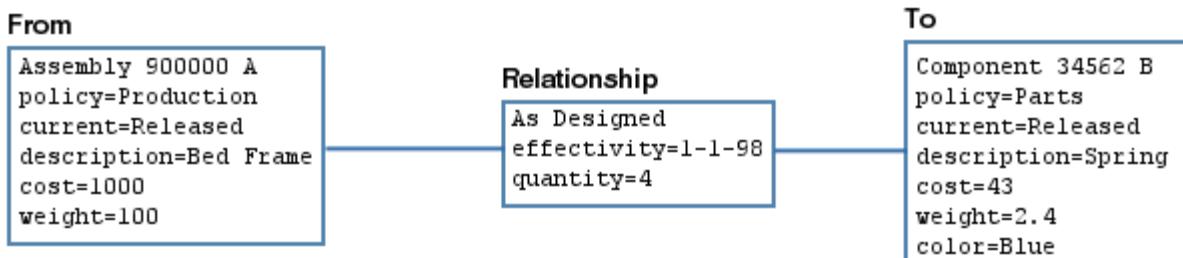
Relationship direction can be a confusing concept when used in `select` expressions. The confusion arises because relationships are also objects, and, during a select navigation, they are treated just like business objects. The direction terms (`to` and `from`) yield different results depending on the point-of-view of their use. If used from a business object, they have exactly the same meaning as in the filter in the Matrix Navigator browser. They mean "navigate to the relationships whose from/to end is connected to this business object."

The following expression means go to the RELNAME relationship whose `from` end is connected to the object:

```
from.relationship[RELNAME]
```

Having reached the relationship, it is possible to get information, such as an attribute value.

Suppose the Assembly is connected with the relationship, As Designed, to another object (Component) and that you want to use the Effectivity attribute in the relationship.



Example 1

The table below shows how this information is obtained with the MQL print command and also how it could be used in finds, tables, and so on.

	MQL print Command	Find or Visuals where clause	Table, Report, Form Field
Select Expression	print bus Assembly 9000 A select from [As Designed]. attribute [effectivity];	Type == Assembly && from[As Designed]. attribute [effectivity] >12-31-99	from[As Designed]. attribute [effectivity]
Result	MQL returns: attribute [effectivity] = 1-1-99	Assembly 90000 A is found, and if in a cue, the visuals are applied.	If Assembly 90000 A is in the table/report/form, 1-1-99 is output in that column/field.

Example 2:

Suppose there are, in fact, many objects connected to the Assembly and you want only Component 34562 B. In this case, you need to also specify the Component. In this example (from the relationship viewpoint), `from` and `to` refer to the business objects at the ends of the relationship. So, in the following expression, `from` refers to a relationship and `to` refers to a business object:

```
from.relationship[As Designed].to
```

For example:

	MQL print Command	Find or Visuals where clause	Table, Report, Form Field
Select Expression	print bus Assembly 9000 A select from [As Designed].to;	from[As Designed]. to.type == Component && from [As Designed]. to.name == 34562 && from[As Designed]. to.rev == B	from [As Designed].to
Result	MQL returns all objects (including Component 34562 B) that are connected to Assembly 90000 A with the relationship As Designed at the <code>to</code> end.	Assembly 90000 is found and, if in a cue, the visuals are applied.	If Assembly 90000 A is in the table/report/form, Component 34562 B is output in that column/field.

Example 3:

Another example finds an attribute value in the business object on the `to` end:

	MQL print Command	Find or Visuals where clause	Table, Report, Form Field

Select Expression	<pre>print bus Assembly 90000 A select from [As Designed] .to.attribute[cost].</pre>	<pre>from[As Designed].to .attribute[cost] < 100 (or >= 40, etc.)</pre>	<pre>from [As Designed].to.attribute[cost]</pre>
Result	MQL returns: 43	Assembly 90000 A is found, and if in a cue, the visuals are applied.	If Assembly 90000 A is in the table/report/form, 43 , is output in that column/field.

Selectable Fields for Applications

This section lists the selectable fields for application objects.

Selectable Fields for Applications

Field	Description	Output
object[].*	A corresponding business object	
description	Application description	Value
id	Application identifier	Value
hidden	Is the application defined as hidden?	True/False
member[].*	Members of the application	Types/Relationships/Attributes
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Application name	Value
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
version	Additional information about the application	Value

Selectable Fields for Attribute

This section lists selectable fields for attribute objects.

Selectable Fields for Attribute

Field	Description	Output
access	The category of access	Private/Protected/Public
application	Application owning this relationship	Application name
default	Default	Returns default value
description	?	Value
hidden	Is this attribute defined as hidden?	True/False
id	Attribute identifier	Value
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
multiline	Is this attribute defined as multiline?	True/False
name	Attribute name	Value
object[]	A corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the attributes's name. If no string is given, the object type is attribute, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
range[].*	?	If range[] is specified, returns that range string, If blank, returns all range strings.
trigger[]	Any trigger configured for the attribute	Returns list of triggers in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARG\$) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not

		configured as a trigger. If you do not include a <code>PROGRAM_NAME</code> , all triggers are listed.
type	Data type value	Returns one of: <code>string</code> , <code>real</code> , <code>integer</code> , <code>timestamp</code> , <code>boolean</code>

Selectable Fields for Business Objects

This section lists selectable fields for business objects.

Selectable Fields for Business Objects		
Field	Description	Output
altowner1	Alternate owner 1	Returns the altowner1 for the business object
altowner2	Alternate owner 2	Returns the altowner2 for the business object
attribute[].*	Attribute instance	If attribute[name] is specified, then default is value; if not, default is a list of all associated attribute names separated by commas.
attribute.inputvalue	The value for the attribute as it was entered by the user, regardless of the normalized value	Returns the input value as it was entered
attribute.inputunit	If the value was entered with no units specified, this selectable returns the default unit.	Returns the input unit as it was entered
attribute.dbvalue	Attribute[ATTRNAME] and attribute [ATTRNAME].value also return the normalized value.	Returns the default (normalized) value
attribute.dbunit	The defined default units for the dimension	Returns the default (normalized) unit
attribute.relationship.*		Not enabled
attribute.rule[].*		Name of access rule that governs the attribute
attribute.type.*		Not enabled
attribute.unitvalue[]	Unitvalue[UNITNAME] is not a stand-alone selectable because it would simply return UNITNAME. It has subselects as defined below.	Returns the converted value in the unit UNITNAME.
##The following are valid for attribute.inputvalue, attribute.dbvalue, and attribute.unitvalue, where * is input, db, or unit:		
attribute.*value.tostring	Converts the specified value to a text string including the values and units	Returns "attribute.*value attribute.*unit"
attribute.*value.systemvalue[SYSTEMNAME]		Returns the unit in the system SYSTEMNAME corresponding to the unit selected off of the attribute.
attribute.*value.systemvalue[SYSTEMNAME].tostring	Converts the specified value to a text string including the values and units	Returns the value and unit of the system SYSTEMNAME.
##The following is valid for attribute.inputunit, attribute.dbunit, and attribute.unitunit (where * is input, db, or unit):		
attribute.*unit.systemunit[SYSTEMNAME]		Returns the value converted to the unit in the system SYSTEMNAME corresponding to the unit selected off of the attribute
attribute.value	Attribute value on this instance	Value
context.*	Context identifier	Returns current user context information
context.role[]	Context role	Returns the name of the context role. With [], returns TRUE/FALSE, depending on whether the string in [] matches the context role.

current.*	Current State	State name (see also Selectable Fields for Policy)
##The following are valid for both <code>current</code> and <code>state</code> :		
current/state.access	Allowed accesses for the current user.	Returns a string of all permissions (with comma separator) for current user
current/state.actual	Actual date	Value
current/state.actual.generic	Actual date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
current/state.current	Is current state?	True/False
current/state.disabled	Is disabled?	True/False
current/state.enabled	Is enabled?	True/False
current/state.name	State name	Value
current/state.overridden	Is overridden?	True/False
current/state.requirements.*	Requirements and accesses on the state	State name
current/state.requirements.access[].*	Accesses for defined users	Returns a string of all permissions (with comma separator)
current/state.requirements.action	Program	Program name
current/state.requirements.autopromote	Can be autopromoted?	True/False
current/state.requirements.check	Program	Program name
current/state.requirements.checkouthistory	Is checkout history enabled?	True/False
current/state.requirements.filter[].*	Is the signature requirement satisfied?	True/False
state.requirements.id	Requirements identifier	Value
current/state.requirements.name	State name	Value
current/state.requirements.notify	Notify message	Value
current/state.requirements.owneraccess	Accesses for owner use	Returns a string of all permissions (with comma separator)
current/state.requirements.publicaccess	Accesses for public use	Returns a string of all permissions (with comma separator)
current/state.requirements.revisionable	Can be revised?	True/False
current/state.requirements.route	Person/group to reassign	User name
current/state.requirements.signature[].*	Signature instance	Name of signature or signer - See state.signature expansion below and in policy state.
current/state.requirements.versionable	Does have versions?	True/False
current/state.revisionable	Is revisionable?	True/False
current/state.satisfied	Are requirements satisfied?	True/False
current/state.scheduled	Scheduled date	Value
current/state.scheduled.generic	Scheduled date in generic format	Value

	(independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	
current/state.signature[].*	Signature instance	Signature properties and accesses
current/state.signature.approved	Approved?	True/False
current/state.signature.comment	Comments from the signer	Comments
current/state.signature.hasapprove	Has been approved?	True/False
current/state.signature.hasignore	Has been ignored?	True/False
current/state.signature.hasreject	Has been rejected?	True/False
current/state.signature.ignored	Ignored?	True/False
current/state.signature.name	Signature name	Signature name
current/state.signature.rejected	Has been rejected?	True/False
current/state.signature.satisfied	Signature requirement is satisfied?	True/False
current/state.signature.signed	Signed?	True/False
current/state.signature.signer.*	Signer ID	Signer ID
current/state.versionable	Is versionable?	True/False depending on how set.
current/state.start	The date on which the object first entered the state.	Date value. Value is an empty string if the object has never been in the state. This differs from the actual date in that the actual date is modified every time the object is promoted or demoted into the state.
current/state.start.generic	The date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) on which the object first entered the state.	Date value. Value is an empty string if the object has never been in the state. This differs from the actual date in that the actual date is modified every time the object is promoted or demoted into the state.
current/state.end	Date on which the object was last in the given state.	Date value. Value is an empty string if the object has never been in the state.. Value is 'NOW' for the current state.
current/state.end.generic	Date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) on which the object was last in the given state.	Date value. Value is an empty string if the object has never been in the state.. Value is 'NOW' for the current state.
current/state.duration	Time spent in the state.	Value is the number of seconds spent in the

		state. If an object is in a state more than once, this is a cumulative value. Value is an empty string if the object has never been in the state.
default.*	Default format for the definition object	Name of Default Format (see Selectable Fields for Formats)
##The following are valid for both <code>default</code> and <code>format</code>		
default/format.businessobject.*	Default format definition object	Name of business object
default/format.file[].*	What files are present?	If file name is specified in [], returns True if that file is associated with the format instance. Otherwise, returns False. If file name is not specified, returns list of all files associated with the format instance.
default/format.file.capturedfile.*	?	Captured file name
default/format.file.format.*	Format object instance	(see Selectable Fields for Formats)
default/format.file.host	Host machine from which the file was checked in.	Host name
default/format.file.location.*	Location object where file is located.	(see Selectable Fields for Location)
default/format.file.name	Name of checked-in file.	Name
default/format.file.obsolete.*	Returns all locations holding a copy of the file that has been marked as obsolete (needing synchronization), regardless of user's site preference.	List of locations
default/format.file.path	Path from which file was checked in.	Path With Version 9 and higher, the output from the format.file.path selectable has changed to use forward slashes (/) instead of back slashes (\). For example, c:\mydocs\testfile.txt now appears as /c/mydocs/testfile.txt.
default/format.file.size	File size in Mb	Value(s)
default/format.file.store.*	Store object where file is located.	(see Selectable Fields for Store)
default/format.file.synchronized.*	Returns all locations holding synchronized versions of the checked in file(s), regardless of user preference.	List of locations.
default/format.format.*	Format definition object	Name of Format (see Selectable Fields for Formats)
default/format.hasfile	Is a file present?	True/False
default/format.modified	Returns the last checkin dates of specified filenames in the specified format.	Last check in dates.
description		Value
expression[NAME]	Evaluate the named expression against the business object	Value
evaluate[STRING]	Evaluate the specified expression string against the business object	Value
exists	Used with no other selectable; Asks: Does this object exist?	Returns true/false if it is the only selectable entered. Otherwise an error occurs.
first.*	Revisions of an object	Returns the revision identifier of the requested object. Works the same as in the

		next level in revisions[].
format[].*	Instances of all format objects	If a name is specified, returns True if an instance of format name actually exists and False if not; if nothing or a pattern is in [] then returns all format type names allowed.
format2[]file.*	Locations where files are synchronized, obsolete or none.	If a name is specified, returns True if an instance of format name actually exists and False if not; if nothing or a pattern is in [] then returns all format type names allowed. When it contains multiple values, lists values in square brackets. Use format2.file.synchronized to identify the locations where files are synchronized. Use format2.file.needsynchronization to identify files needing synchronization from one store/location to another store/location. Returns True or False. Use format2.file.status to display status of the file. Returns store and all locations in brackets and displays status of file: synchronized, obsolete or none.
from[].*	<i>from</i> relationships	If the specified relationship exists returns True; otherwise False. If nothing or a pattern is specified in [], lists names of all the associated relationships.
grant.*	[??]	Value
grantee.*	User(s) to whom access is granted	Value
granteeaccess	Grantee accesses	Value
granteesignature	Have signature accesses been granted?	True/False
grantkey	Grant identifier	Value
grantor.*	User granting access	Value (see Selectable Fields for User)
history.*	Text log of history	If history[n] = returns the nth history entry. If nothing is specified in [], returns log of all entries.
history.approve	Text log of history	Returns a list of approve history records
history.changename	Text log of history	Returns a list of changename history records
history.changeowner	Text log of history	Returns a list of changeowner history records
history.changepolicy	Text log of history	Returns a list of changepolicy history records
history.changetype	Text log of history	Returns a list of changetype history records
history.changevault	Text log of history	Returns a list of changevault history records
history.checkin	Text log of history	Returns a list of checkin history records
history.checkout	Text log of history	Returns a list of checkout history records
history.connect	Text log of history	Returns a list of connect history records
history.create	Text log of history	Returns a list of create history records
history.custom	Text log of history	Returns a list of custom history records
history.demote	Text log of history	Returns a list of demote history records
history.disable	Text log of history	Returns a list of disable history records
history.enable	Text log of history	Returns a list of enable history records

history.grant	Text log of history	Returns a list of grant history records
history.ignore	Text log of history	Returns a list of ignore history records
history.lock	Text log of history	Returns a list of lock history records
history.modify	Text log of history	Returns a list of modify history records
history.modifyattribute	Text log of history	Returns a list of modifyattribute history records
history.movedoid	Text log of history	Returns a list of movedoid history records
history.override	Text log of history	Returns a list of override history records
history.promote	Text log of history	Returns a list of promote history records
history.purge	Text log of history	Returns a list of purge history records
history.reject	Text log of history	Returns a list of reject history records
history.removedoid	Text log of history	Returns a list of removedoid history records
history.removefile	Text log of history	Returns a list of removefile history records
history.revise	Text log of history	Returns a list of revise history records
history.revoke	Text log of history	Returns a list of revoke history records
history.schedule	Text log of history	Returns a list of schedule history records
history.state	Text log of history	Returns a list of all states the object was in when operations were performed on it.
history.time	Text log of history	Returns a list of the timestamps of every history record.
history.time.generic	Text log of history	Returns a list of the timestamps in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) of every history record.
history.unlock	Text log of history	Returns a list of unlock history records
history.user	Text log of history	Returns a list of all users who have operated on the object
history.between[FROMDATE TODATE]	Subset of entire history log	<p>With no dates included, returns a list of all history records.</p> <p>When dates are included, they must adhere to the date format of the system.</p> <p>If both FROMDATE and TODATE are provided, the list includes events that occurred between the 2 dates, inclusive of FROMDATE but exclusive of TODATE.</p> <p>If only FROMDATE is included, the list includes events that occurred between the date specified and the present date, inclusive of both FROMDATE and the present date.</p> <p>If only TODATE is specified, the list includes events that occurred before the date specified, exclusive of TODATE.</p>
history.count[FROMDATE TODATE]	History log count	<p>With no dates included, returns an integer which is the total number of history records for the object.</p> <p>When dates are included, they must adhere to the date format of the system.</p> <p>If both FROMDATE and TODATE are provided, the list includes events that occurred between the 2 dates, inclusive of FROMDATE but exclusive of TODATE.</p>

		If only FROMDATE is included, the list includes events that occurred between the date specified and the present date, inclusive of both FROMDATE and the present date. If only TODATE is specified, the list includes events that occurred before the date specified, exclusive of TODATE.
history.description	List of history log details	Returns a list of entries for each history record, that includes any fields other than state, user, event, or date concatenated together. Differs depending on the event.
history.description.between[FROMDATE TODATE]	List of history log details	When between.dates are included, they must adhere to the date format of the system. If both FROMDATE and TODATE are provided, the list includes events that occurred between the 2 dates, inclusive of FROMDATE but exclusive of TODATE. If only FROMDATE is included, the list includes events that occurred between the date specified and the present date, inclusive of both FROMDATE and the present date. If only TODATE is specified, the list includes events that occurred before the date specified, exclusive of TODATE.
history.event	List of history events	Returns a list of all history events that have occurred for the object.
history.user.between[FROMDATE TODATE]	List of users	Returns the user responsible for each history record. When between.dates are included, they must adhere to the date format of the system. If both FROMDATE and TODATE are provided, the list includes events that occurred between the 2 dates, inclusive of FROMDATE but exclusive of TODATE. If only FROMDATE is included, the list includes events that occurred between the date specified and the present date, inclusive of both FROMDATE and the present date. If only TODATE is specified, the list includes events that occurred before the date specified, exclusive of TODATE.
history.time.between[FROMDATE TODATE]	List of history event times	Returns the timestamps of history events. When between.dates are included, they must adhere to the date format of the system. If both FROMDATE and TODATE are provided, the list includes events that occurred between the 2 dates, inclusive of FROMDATE but exclusive of TODATE. If only FROMDATE is included, the list includes events that occurred between the date specified and the present date, inclusive of both FROMDATE and the present date. If only TODATE is specified, the list includes events that occurred before the date specified, exclusive of TODATE.
history.time.between[FROMDATE TODATE].generic	List of history event times	Returns the timestamps in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) of history events. When between.dates are included, they must adhere to the date format of the system. If both FROMDATE and TODATE are provided,

		<p>the list includes events that occurred between the 2 dates, inclusive of FROMDATE but exclusive of TODATE.</p> <p>If only FROMDATE is included, the list includes events that occurred between the date specified and the present date, inclusive of both FROMDATE and the present date.</p> <p>If only TODATE is specified, the list includes events that occurred before the date specified, exclusive of TODATE.</p>
id	Object identifier	Value
islockingenforced	Is checkin allowed only if the object is locked?	True if enforced locking is turned on. False if not.
last.*	Revisions of an object	Returns the revision identifier of the requested object. Works the same as in the next level in revisions[].
lattice.*	Vault name	Value (see Selectable Fields for Vault)
locked	Is this object locked?	True/False
locker.*	Administrative object	User that Locked object
logicalid	<p>A business object identifier. A logical id is a global identifier shared by all members of a revision sequence.</p> <p>This selectable is only available if your database has been upgraded to use physical and logical IDs.</p>	Value
method.*	?	Returns all programs that are associated as methods
modified	Modification time and date	Value
modified.generic	Modification time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
name	Business object name	Value
next.*	Revisions of an object	Returns the revision identifier of the requested object. Works the same as in the next level in revisions[].
originated	Creation time and date	Value
originated.generic	Creation time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
owner.*	?	Value (see Selectable Fields for User)
policy.*	?	Value (see Selectable Fields for Policy)
physicalid	<p>A business object identifier. A physical id is a global identifier that is unique to each business object</p> <p>This selectable is only available if your database has been upgraded to use physical and</p>	Value

	logical IDs.	
previous.*	Revisions of an object	Returns the revision identifier of the requested object. Works the same as in the next level in revisions[].
program.*	?	value (see Selectable Fields for Program)
relationship[].*	All relationship instances	If the specified relationship exists returns True; otherwise False. If nothing or a pattern is specified in [], lists names of all existing relationships.
relationship.attribute[].*	Attributes of the relationship	Lists attributes of relationship if nothing specified in []. With a value in [] returns attribute value. (see Selectable Fields for Attribute)
relationship.businessobject.*	The "other" side of the relationship	Returns to object. If used with ".to.relationship", returns name of <i>from</i> object.
relationship.context.*	Context identifier	Returns current context user.
relationship.from.*	Business object on <i>from</i> side	Name of <i>from</i> associated business object.
relationship.history.*	Text log of history	If history [n] = returns the nth history entry. If nothing is specified in [], returns log of all entries.
relationship.id	Relationship identifier	Value
relationship.isfrozen	Frozen relationship?	Returns list of all relationships with True for frozen, False for not frozen.
relationship.method	Relationship method	Returns the name of the relationship ID
relationship.name	Relationship definition	Relationship name
relationship.originated	Timestamp when relationship was originated	Value
relationship.propagatemodifyfrom	Is propagatemodify enabled on from businessobject?	True/False
relationship.propagatemodifyto	Is propagatemodify enabled on to businessobject?	True/False
relationship.rule[].*	Relationship rule that governs that relationship	Returns the name of the relationship rule for that relationship
relationship.to.*	Business object on <i>to</i> side	Name of <i>to</i> associated business object.
relationship.type.*	Relationship definition	Relationship type
reserved	Boolean indication reserved status of a business object	True/False
reservedby	Non-empty string or the context user	Name of the person who reserved the object.
reservedstart	The date or timestamp of when the business object was reserved.	Returns the date and time of when that object was reserved.
reservedstart.generic	The date or timestamp in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) of when the business object was reserved	Returns the date and time of when that object was reserved.
reservedcomment	Optional comment not exceeding 254 characters from the person	Any comments entered.

	reserving the object.	
revision	Business object revision	Value
revisions[].*	Existing revisions of an object	If [] are left blank, returns the revision identifiers that are in use. If a revision is specified, returns True if it exists, False if not.
search.*		True/False
search.author		Evaluates to an optional author returned by the search engine
search.capturedfile.*		True/False
search.format.*		(see Selectable Fields for Formats)
search.host		Host name
search.location.*		(see Selectable Fields for Location)
search.name		Name
search.path		Path name
search.score		Evaluates to an optional search score returned by the search engine
search.size		Size
search.store.*		(see Selectable Fields for Store)
search.title		Evaluates to an optional document title returned by the search engine.
search.url		Returns the url address
state[].*	States used in the lifecycle of the policy	Returns state names from lifecycle
to[].*	<i>to</i> relationships	If the specified relationship exists returns True; otherwise, False. If nothing or a pattern is specified in [], lists names of all the associated relationships.
trigger[]	any trigger configured for the object	Returns list of triggers in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARGS) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not configured as a trigger. If you do not include a PROGRAM_NAME, all triggers are listed.
immediatetrigger[]	any trigger configured for the object; does not include inherited triggers	Returns list of triggers directly configured for the type in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARGS) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not configured as an immediatetrigger. If you do not include a PROGRAM_NAME, all immediatetriggers are listed.
type.*	<i>to</i> relationships	If the specified relationship exists returns True; otherwise, False. If nothing or a pattern is specified in [], lists names of all the associated relationships.
unit[UNITNAME].default	Use to determine whether or not	Returns TRUE or FALSE indicating whether or

	the specified unit is the normalized value for the dimension	not the unit is the default (normalized) unit
unit[UNITNAME].multiplier	Use to determine the multiplier value for the specified unit	Returns the value of the multiplier used to convert data from the input value to the normalized value
unit[UNITNAME].offset	Use to determine the offset value for the specified unit	Returns the value of the offset used to convert data from the input value to the normalized value
unit[UNITNAME].label	Use to retrieve the label assigned to the unit	Returns the label of the unit
unit[UNITNAME].dimension	Use to retrieve the name of the dimension for which the unit is defined	Returns the dimension that the unit is part of
unit[UNITNAME].systemunit	Use to determine the system to which the unit belongs	If an index (systemname) is given, returns the corresponding systemunit. If no index is given, returns a list of all systemunits applied to this unit.
vault.*	Vault definition	Vault name (see Selectable Fields for Vault)
workflow[].*		(See Selectable Fields for Query)

Selectable Fields for Commands

This section lists selectable fields for commands.

Selectable Fields for Commands		
Field	Description	Output
alt	Command's alt	Value
code	Command's associated JavaScript code	Value
description	Command object description	Value
global	Is this command available to all users?	True/False
hidden	Is this command defined as hidden?	True/False
href	Command's href	Value
id	Command identifier	Value
label	Command's label	Value
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Command object name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the command's name. If no string is given, the object type is command, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
setting[].*	Command's settings	Returns a list of all setting names. Use setting.value to get a list of name value pairs, or specify setting[].value to get one value.
type	Administrative object type	Command
user[].*	Command's users	Returns all user names.

Selectable Fields for Connections

This section lists the selectable fields for connections.

Selectable Fields for Connections

Field	Description	Output
altowner1	Alternate owner 1	Returns the altowner1 for the connection
altowner2	Alternate owner 2	Returns the altowner2 for the connection
attribute[].*	Attribute values on the connection	If an attribute is specified default is attribute value. If nothing is specified in [], returns attributes on the relationship type.
businessobject.*	End objects of the connection.	Returns type, name, and revision of objects on both ends of the relationship.
context.*	Context identifier	Returns current context user.
from.*	<i>from</i> object or relationship	Returns type, name, and revision of objects on from end. If the from end is a business object, returns the business object. If the from end is a relationship, returns null. Previously, this field always returned a business object. To retrieve the from end regardless of its type, use the fromall field.
fromall.*	From side of the connection	Returns the ID of the from side If it is a business object, returns the business object ID prefixed with a B. If it is a relationship, returns the connection ID prefixed with R. It is not necessary to strip off the prefix when passing these ids back into the system for subsequent operations.
frommid.*	The connections pointing from this connection.	List of connections. Supports optional relationship type pattern, for example frommid[EBOM].
fromrel.*	The connection at the from end.	If it is a connection, returns the connection If it is a business object, returns null
history.*	Connection history log file	If history [n] = returns the nth history entry. If nothing is specified in [], returns log of all entries. The first entry is identified as "0".
id	Connection identifier	Value
interface	Name	Name of the interface
isfrozen	Has the connection been Frozen?	True/False
logicalid	A connection identifier. A logical id is a global identifier shared by all members of a revision sequence. This selectable is only available if your database has been upgraded to use physical and logical IDs.	Value
method.*	Unused	

modified	Modification time and date	Value
modified.generic	Modification time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
name	Relationship type or connection name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the connection's name. If no string is given, the object type is connection, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Creation time and date	Value
originated.generic	Creation time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
owner.*	?	Value (see Selectable Fields for User)
physicalid	A connection identifier. A physical id is a global identifier that is unique to each connection. This selectable is only available if your database has been upgraded to use physical and logical IDs.	Value
propagatemodifyfrom	Is propagatemodify enabled on <i>from</i> businessobject.	True/False
propagatemodifyto	Is propagatemodify enabled on <i>to</i> businessobject.	True/False
reserved	Boolean indicator of a reserved status for a connection	True/False
reservedby	Non-empty string or the context user	Name of the person who reserved the connection.
reservedstart	The date or timestamp of when the connection was reserved	Returns the date and time of when that connection was reserved.
reservedstart.generic	The date or timestamp in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) of when the connection was reserved	Returns the date and time of when that connection was reserved.
reservedcomment	Optional comment not exceeding 254 characters from the person reserving the object	Any comments entered.
rule[].*	Rule values on connections	If rule name [] is specified, list accesses allowed by current context user. If no rule name[] is specified, list associated rules
to.*	<i>to</i> object or relationship	Returns type, name, and revision of objects on to end. If the to end is a business object, returns the business object. If the to end of the connection is a connection,

		<p>returns null.</p> <p>Previously, this field always returned a business object.</p> <p>To retrieve the to end regardless of its type, use the toall field.</p>
toall	To side of the connection	<p>Returns the ID of the to side</p> <p>If it is a business object, returns the business object ID prefixed with a B.</p> <p>If it is a relationship, returns the connection ID prefixed with R.</p> <p>It is not necessary to strip off the prefix when passing these ids back into the system for subsequent operations.</p>
tomid.*	The connections pointing to this connection	<p>List of connections</p> <p>Supports optional relationship type patterns, for example, tomid[EBOM].</p>
torel.*	The connection at the to end	<p>If it is a connection, returns the connection</p> <p>If it is a business object, returns null</p>
type.*	Object types that may be connected by the relationship type	Value

Selectable Fields for Dataobjects

This section lists the selectable fields for dataobjects.

Selectable Fields for Dataobjects

Field	Description	Output
name	dataobject name	Value
description	dataobject description	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
hidden	Is this dataobject defined as hidden?	True/False
id	Dataobject identifier	Value
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the dataobject's name. If no string is given, the object type is dataobject, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
user.*	User whose workspace the dataobject is in.	Value
visible[]	List of users that have visibility to the dataobject	Value list
type	Type parameter of the dataobject...not corresponding to TYPE administrative object.	String
value	String value	Value

Selectable Fields for Dimensions

This section lists the selectable fields for dimensions.

Selectable Fields for Dimension

Field	Description	Output
attribute	Lists the attributes to which this dimension has been applied	Returns the name of the attribute(s) this dimension applies to
attribute[ATTRNAME]	Use to determine whether or note a specific attribute has this dimension applied to it	Returns TRUE or FALSE indicating whether or not the dimension is applied to the specified attribute
dbunit	Database unit: the units of the default (normalized) dimension	Returns the name of the default unit
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the dimension's name. If no string is given, the object type is dimension, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
system	Lists the systems associated with the dimensin	Returns the systems defined in the dimension
system[SYSTEMNAME]	Use to determine whether or note a specific system name has been defined for this dimension	Returns TRUE or FALSE indicating whether or not the system is applied to the dimension
unit	All units defined for the dimension	Returns the names of the units in the dimension
unit[UNITNAME]	Use to determine whether or not a specific unit has been added to a dimension	Returns TRUE or FALSE indicating whether or not the unit is defined for the dimension
unit[UNITNAME].default	Use to determine whether or not the specified unit is the normalized value for the dimension	Returns TRUE or FALSE indicating whether or not the unit is the default (normalized) unit
unit[UNITNAME].multiplier	Use to determine the multiplier value for the specified unit	Returns the value of the multiplier used to convert data from the input value to the normalized value
unit[UNITNAME].offset	Use to determine the offset value for the specified unit	Returns the value of the offset used to convert data from the input value to the normalized value
unit[UNITNAME].label	Use to retrieve the label assigned to the unit	Returns the label of the unit
unit[UNITNAME].dimension	Use to retrieve the name of the dimension for which the unit is defined	Returns the dimension that the unit is part of
unit[UNITNAME].systemunit	Use to determine the system to which the unit belongs	If an index (systemname) is given, returns the corresonding systemunit. If no index is given, returns a list of all systemunits applied to this unit.

Selectable Fields for Expressions

This section lists selectable fields for expressions.

Selectable Fields for Expressions

Field	Description	Output
description	Expression object description	Value
hidden	Is this expression defined as hidden?	True/False
id	Expression identifier	Value
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Expression object name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the expression's name. If no string is given, the object type is expression, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
value	The expression string	Value

Selectable Fields for Form Fields

For a list of selectables for fields in a form, enter `print form selectable field;` in MQL. Selectables for properties related to geometry and display apply only to non-Web forms. Selectables for properties related to URL links, settings, and user access apply to Web forms only.

Selectable Fields for Form Fields

Field	Description	Output
field.description	Field description	Value
field.property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
field.hidden	Is this form field defined as hidden?	True/False
field.id	Field identifier	Value
field.user[].*	User assigned to field	If user [] is specified, returns assigned user for field. If blank, returns assigned user for each field.
field.modified	Time and date the field was modified	Value
field.originated	Time and date the field was created	Value
field.type	Type of field	Field type
field.number	Number assigned to field based on its creation order in form	Non-Web forms: label, select, graphic, image, icon Web forms: select
field.label	Field label	Non-Web forms: null (labels are separate fields) Web forms: label value
field.value	Synonym for field label	Non-Web forms: null (labels are separate fields Web forms: label value
field.selectedvalue	[Unused]	
field.expression	Expression (select statement) associated with field	Type of selectable item (attribute, description, etc.)
field.expressiontype	Type of object expression applies to	Business object or relationship
field.font +	Font used to display field information	Value
field.editable	Is the field editable?	True/False
field.multiline	Can field display more than one line?	True/False
field.minwidth +	Minimum width of field	Value
field.minheight +	Minimum height of field	Value
field.autowidth	Does the field width autosize?	Non-Web forms: True/False. Web forms: True
field.autoheight	Does the field height autosize	Non-Web forms: True/False. Web forms: True
field.absolutex +	Is horizontal location value absolute?	True/False
field.absolutey +	Is vertical location value absolute?	True/False
field.xlocation +	Horizontal location of field (left side)	Value
field.ylocation +	Vertical location of field (top)	Value
field.width +	Width of field in unit specified for field	Value
field.height +	Height of field in unit specified for field	Value

field.href ‡	Field's href	URL plus parameters
field.alt ‡	Field's alt	Alt text
field.range ‡	JSP that gets a range of values and populates field with selected value.	JSP name
field.update ‡	Page to display after field is updated	URL
field.setting[].[*] ‡	Associated settings	If setting [] is specified, returns setting names and values. If blank, returns all setting names and values.

+ Available only for standard, non-Web forms ‡ Available only for Web forms

Selectable Fields for Formats

Selectable Fields for Format		
Field	Description	Output
description	?	Value
edit	Program	Program name
filesuffix	Suffix used for files of this format	Value
hidden	Is this Format defined as hidden?	True/False
id	Format identifier	Value
mime	Mime type associated with the format	value. type and subtype separated by a slash. For example, <code>text/plain</code> or <code>text/jsp</code> .
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Format name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the format's name. If no string is given, the object type is format, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
print	Program	Program name
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
version	Version of format software	Value
view	Program	Program name

Selectable Fields for Forms

Selectables for properties related to geometry and display apply only to non-Web forms.

Selectable Fields for Forms

Field	Description	Output
background +	Background color	Value
description	Form object description	Value
editorunits +	The unit chosen for the form's measurements	picas, points, or inches
field[].*	Form fields	If field [] is specified, returns field's subfields. If blank, returns selectable subfields for all fields.
footer +	Footer height in unit specified for form	Value
foreground +	Foreground color	Value
header +	Header height in unit specified for form	Value
height +	Height of form in unit specified for form	Value
hidden	Is this form defined as hidden?	True/False
id	Form identifier	Value
leftmargin +	Left margin width in unit specified for form	Value
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Form object name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the form's name. If no string is given, the object type is form, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and

		values. If blank, returns all property names and values.
rightmargin +	Right margin width in unit specified for form	Value
type[].*	Types explicitly assigned to the form. Does not include derived types	Types assigned to the form
web	Is this form a Web form?	True/False
width +	Width of form in unit specified for form	Value

+ Available only for standard, non-Web forms.

Selectable Fields for Groups

This section lists the selectable fields for groups.

Selectable Fields for Group

Field	Description	Output
admin	?	Not enabled
ancestor[].*	Ancestors of the group	Returns a list of all ancestors of the current group. With [], returns TRUE/FALSE, depending on whether any ancestor matches the condition in []. If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string is considered the pattern.
assignment[].*	Assignments that the group has	Lists all assignments
child[].*	Child values on group	If child[] is specified, returns child names and values. If blank, returns all child names and values.
description	?	Value
email	?	Value
hidden	Is this group defined as hidden?	True/False
id	Group identifier	Value
isingroup	Is a group?	True/False
isaperson	Is a person?	True/False
isarole	Is a role?	True/False
mask	List of accesses	Defaults to accesses of person
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Group name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the group's name. If no string is given, the object type is group, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and	Value

	MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	
parent[].*	Parents of the role and group.	<p>Without any "." or [], returns a list of all parents. With [], returns TRUE/FALSE, depending on whether any parent matches the condition in [].</p> <p>If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string becomes the pattern.</p>
person.*	Person values on group	If person[] is specified, returns person names and values. If blank, returns all person names and values (see Selectable Fields for Person).
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.

Selectable Fields for Inquiry

This section lists the selectable fields for inquiry objects.

Selectable Fields for Inquiry		
Field	Description	Output
name	Inquiry object name	Value
description	Inquiry object description	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
hidden	Is this inquiry defined as hidden?	True/False
id	Inquiry identifier	Value
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the inquiry's name. If no string is given, the object type is inquiry, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
code	Defined code	Value
pattern	Defined pattern	Value
format	Defined format	Value
argument[].*	Specified arguments	Value

Selectable Fields for Interface

This section lists the selectable fields for interface objects.

Selectable Fields for Interface		
Field	Description	Output
abstract	Is this interface defined as abstract?	True/False
allchildren	All child interfaces	Returns names of all child interfaces. Same as derivative.
allparents	All parent interfaces	Names of all parents.
attribute[].*	Attributes of interface	If [name] is specified, returns True if name is an attribute of the interface; False if not. If no name is specified, returns all attributes.
derivative[].*	All child interfaces	If [name] is specified, returns True if name is child of this object; False if not. If no name is specified, returns names of all child interfaces (same as allparents).
derived.*	Immediate parent interfaces	Name of the immediate parent interface definition
description	Interface object description	Value
hidden	Is this interface defined as hidden?	True/False
id	Interface identifier	Value
immediateattribute[].*	Non-inherited attributes	If [name] is specified, returns True if name is an immediateattribute of the interface; False if not. If no name is specified, returns all immediateattributes.
immediatederivative[].*	Immediate child interfaces	Returns first level children, see derivative for defaults.
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Interface object name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the interface's name. If no string is given, the object type is interface, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the	Value

	administration object was created	
property[].*	Associated properties	If [name] is specified, returns property name and value. If blank, returns all property names and values.
relationship	Name	Names of relationship types.

Selectable Fields for Location

This section lists the selectable fields for locations.

Selectable Fields for Location		
Field	Description	Output
compressed	Is location compressed?	True/False
deletetrigger	The name of the java class that implements com.matrixone.fcs.mcs.DeleteTrigger interface.	Value
description	Location description	Value
encrypted	Is location encrypted?	True/False
hashed	Is this location hashed?	True/False
hidden	Is this location defined as hidden?	True/False
host	Location of store	Value
id	Location identifier	Value
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Location name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the location's name. If no string is given, the object type is location, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
path	Path of store	Value
permission	Read/write permissions for owner group world	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.

protocol	Protocol of captured location.	One of ftp, ftps, file, or nfs.
search[].*	Used for URL-based full text search	If search[] is specified, returns information about Product?Line files matching query; otherwise, returns filenames.
size.[UNIT]	Used with hard link for FCS local clone feature (see <i>File Collaboration Server > Large Files > Hard Link for Local Clone</i>). Possible values for UNIT are: <ul style="list-style-type: none"> • byte • mb (megabyte) • gb (gigabyte) 	Total size of the files physically located in a location's path as computed by using ENOVIA V6 file metadata. Obsolete files also count.
url	Page from Web server	Value
prefix	The prefix defined for the location.	Value

Selectable Fields for Menus

This table lists the selectable fields for menus.

Selectable Fields for Menus		
Field	Description	Output
alt	The menu's alt	Value
child[].*	The menu's children	List of all items in the menu. (commands and submenus)
command	The menu's commands	List of all commands in the menu.
description	Menu object description	Value
hidden	Is this menu defined as hidden?	True/False
href	The menu's href	Value
id	Menu identifier	Value
label	Menu's label	Value
menu	The menus' submenus	List of all submenus in the menu
modified	Time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Menu object name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the menu's name. If no string is given, the object type is menu, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
setting[].*	Menu's settings	Returns a list of all setting names. Use setting.value to get a list of name value pairs, or specify setting[].value to get one value.
type	Administrative object type	Menu

Selectable Fields for Person

This section lists the selectable fields for person objects.

Selectable Fields for Person		
Field	Description	Output
address	From person definition	Value
admin	List of administrative accesses	Defaults to administrative accesses of person
application	Default application assigned to the person.	Application name
assignment[].*	Assignments that the person has	<p>Lists all assignments to groups, roles, and associations. Without [], returns a list of all assignments for this person. With [], returns TRUE/FALSE, depending on whether any assignment matches the condition in [].</p> <p>If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string is considered the pattern.</p>
business	Is business user?	True/False
certificate[].*	Security certificate	Value
description		Value
email	Email address	Value
fax	From person definition	Value
full	Is Product?Line user?	True/False
fullname	From person definition	Value
hidden	Is this person defined as hidden?	True/False
id	Person identifier	Value
inactive	Is inactive user?	True/False
isagroup	Is a group?	True/False
isaperson	Is a person?	True/False
isarole	Is a role?	True/False
isassigned	Is assigned to a given group?	True/False
lattice[].*	Vault definition	Vault name (see Selectable Fields for Vault)
ldap	?	
mask	List of accesses	defaults to accesses of person
modified	The time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Person name	Value

neverexpire	Boolean	True or False depending on the setting of neverexpire on the person object.
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the person's name. If no string is given, the object type is person, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
password	The password setting on the person.	The value is one of the following: <DISABLED>, which means the password has been disabled. <RESTRICTED>, which means a password is set. <NONE>, which means the password is set to nopassword.
passwordexpired	Boolean	True or False depending on the setting of passwordexpired on the person object.
phone	From person definition	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
system	Is system user?	True/False
vault[].*	Vault definition	Vault name (see Selectable Fields for Vault)

Selectable Fields for Policy

This section lists the selectable fields for policy objects.

Selectable Fields for Policy		
Field	Description	Output
allstate	Allstate access rule	Returns TRUE/FALSE indicating whether the policy has an allstate access definition.
defaultformat.*	The default Format definition	name of default Format
description	The policy name	Value
format[].*	Format definitions allowed	Allow Format names (see Selectable Fields for Formats)
hidden	Is this policy defined as hidden?	True/False
id	Policy identifier	Value
islockingenforced	Is checkin allowed only if the object is locked?	True, if enforced locking is turned on; False if not.
modified	The time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	The policy name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the policy's name. If no string is given, the object type is policy, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
revision	The revision sequence of policy	Value
state[].*	The states used in the lifecycle of the policy	Returns state names from lifecycle
state.access[]	Accesses for defined users	Returns a list of states that have user accesses defined and the permissions granted to the user.

		For example: state[Planned].access[Product Planner] = all Only states that have user accesses defined are listed. Use state.owneraccess or state.publicaccess for other accesses.
state.action	The program	Program name
state.autopromote	Can be autopromoted?	True/False
state.check	The program	Program name
state.checkouthistory	Is checkout history enabled?	True/False
state.filter[].*	Is the signature requirement satisfied?	True/False
state.id	The state identifier	Value
state.name	Accesses for public use	Value
state.notify	The notify message	Value
state.owneraccess	Accesses for owner user	Returns a string of all permissions (with comma separator)
state.publicaccess	Accesses for public use	Returns a string of all permissions (with comma separator)
state.revisionable	Is revisionable?	True/False
state.route	The person/group to reassign	User name
state.signature[].*	?	If signature name is specified, the signer name is returned. Otherwise, lists all signatures.
state.signature.approve[].*	User (person, group, or role)	see Selectable Fields for Person , Selectable Fields for Groups , Selectable Fields for Role
state.signature.hasapprove	Does the current context user has authorization to approve?	True/False
state.signature.hasignore	Does the current context user has authorization to ignore?	True/False
state.signature.hasreject	Does the current context user has authorization to reject?	True/False
state.signature.ignore[].*	User (person, group, or role)	see Selectable Fields for Person , Selectable Fields for Groups , Selectable Fields for Role
state.signature.name	Signature object	Signature name
state.signature.reject[].*	User (person, group, or role)	see Selectable Fields for Person , Selectable Fields for Groups , Selectable Fields for Role
state[].trigger[]	any trigger configured for the policy state	Returns list of triggers in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARGS) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not configured as a trigger. If you do not include a PROGRAM_NAME, all triggers are listed.
state.versionable	Is versionable?	True/False
store.*	The store specified by policy	Returns Store name (see Selectable Fields for Store)
type[].*	The types governed by the policy	Allows type names (see Selectable Fields for Type)

Selectable Fields for Process

This section lists selectable fields for process objects.

Selectable Fields for Process		
Field	Description	Output
attribute[].*	Attribute instance	If attribute[name] is specified, then default is value; if not, default is a list of all associated attribute names.
automated[].*	The automated activity	If automated[name] is specified, then default is value; if not, default is a list of all associated automated activity names (see Selectable Fields for Query)
automated[].trigger[]	any trigger configured for the process state	Returns list of triggers in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARGS) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not configured as a trigger. If you do not include a PROGRAM_NAME, all triggers are listed.
autostart	Is autostart enabled?	True/False
context.*	The context identifier	Returns current context user.
description	The process description	Value
hidden	Is Process defined as hidden?	True/False
id	The workflow identifier	Value
interactive[].*	The interactive activity	If interactive[name] is specified, then default is value; if not, default is a list of all associated interactive activity names. (see Selectable Fields for Query)
interactive[].trigger[]	any trigger configured for the process state	Returns list of triggers in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARGS) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not configured as a trigger. If you do not include a PROGRAM_NAME, all triggers are listed.
modified	The time and date the administration object was modified	Value
modified.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	The workflow name	Value
node[].*	The element of process	If node is specified, default is value. If nothing is specified in [], returns node names for process.
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the process's name. If no string is given, the object type is process, with case ignored. In a case sensitive

		environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was created	Value
originated.generic	Time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
transition	Transition condition	Value
trigger	Trigger program definitions	Value (see Selectable Fields for Program)
trigger[]	any trigger configured for the process state	Returns list of triggers in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARGS) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not configured as a trigger. If you do not include a PROGRAM_NAME, all triggers are listed.

Selectable Fields for Product

This section lists the selectable fields for product objects.

Selectable Fields for Product		
Field	Description	Output
addon	Is the product an add-on product?	True/False
description	Product description.	Value
dependency	Relationship to licensed products.	Value
derivative	Relationship to licensed products	
dynamic	Is the product a dynamic product configuration?	True/False
hidden	Is product defined as hidden?	True/False
modified	The time and date the product was modified	Value
modified.generic	Modification time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
originated	The time and date the product was created	Value
originated.generic	Creation time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
person[].*	Persons assigned to the product.	Without any "." or [], returns a list of all persons. With [], returns TRUE/FALSE, depending on whether any person matches the condition in []. If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string becomes the pattern.
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
richclient	Is the product a rich client?	True/False
technical	Is the product a technical product?	True/False
title	The full name of the dynamic product configuration.	Value
webclient	Is the product a web client?	True/False

Selectable Fields for Program

This section lists the selectable fields for program objects.

Selectable Fields for Program		
Field	Description	Output
classname	The code's classname (JPO only).	For Java programs only, the classname of the code.
code	Program code	For MQL or Tcl, lists the actual code; for external program, lists the command to launch the program.
description	Program description	Value
doesneedcontext	Does this program require a business object?	True/False
doesuseinterface	Is this program defined as launchable from a collaborative server application? (deferred and downloadable)	True/False
downloadable	Does program include code for operations not supported on the Web product?	True/False
execute	How is the execute flag set?	deferred or immediate
hidden	Is this program object hidden?	True/False
id	Program identifier	Value
isafunction	Is program used as a function? (trigger, state check/action, format edit/view/print, frame prologue/epilogue, widget load/validate, button command)	True/False
isamethod	Is this program associated with a type?	True/False
isjavaprogram	Is this program type Java	True/False
ismqlprogram	Is this program type MQL?	True/False
ispipedprogram	Is this program type piped	True/False
iswizardprogram	Is this program type a wizard	True/False
modified	The time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	The program name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the program's name. If no string is given, the object type is program, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest

		originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was originated	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was originated	Value
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
type[].*	Includes all types which are derived from types specifying the program as a method (that is, all types for which the program is an inherited method)	Type for which the specified program is a method.
user	The associated user	Returns user name.

Selectable Fields for Query

This section lists the selectable fields for query objects.

Selectable Fields for Query		
Field	Description	Output
description	Not used	
expandtype	Find types that are children of types searched for?	True/False
hidden	Is this query defined as hidden?	True/False
id	Query identifier	Value
modified	The time and date the query was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the query was modified	Value
name	Query name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the query's name. If no string is given, the object type is query, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the query was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the query was created	Value
owner	Value of owner field for query	value (see Selectable Fields for User)
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
revision	Revisions to be included in search criteria.	Value
type	Types to be included in search criteria.	
vault	Vaults to be included in search criteria.	vault name (see Selectable Fields for Vault)
where	Where clause of query	Value

Selectable Fields for Relationships

This section lists the selectable fields for relationship objects.

Selectable Fields for Relationship		
Field	Description	Output
abstract	Is this Relationship defined as abstract?	True/False
access	The category of access	Private/Protected/Public
application	Application owning this relationship	Application name
attribute[].*	Attribute instance	If attribute[name] is specified, then default is value; if not, default is a list of all associated attribute names separated by commas.
derivative[].*	All child relationships	If [name] is specified, returns True if name is child of this object; False if not. If no name is specified, returns names of all child relationships.
derived.*	Parent relationship	Name of the parent relationship definition
description	?	Value
fromaction	?	Returns the name of the action rule (None , Float , or Replicate) on the <i>from</i> side of relationship
fromcardinality	<i>from</i> object	Returns <i>from</i> cardinal ID
fromcloneaction	<i>from</i> clone rule	Returns the name of the clone rule (None , Float , or Replicate) on the <i>from</i> side of relationship
frommeaning	<i>from</i> object	Returns <i>from</i> meaning
fromreviseaction	<i>from</i> revision rule	Returns the name of the revision rule (None , Float , or Replicate) on the <i>from</i> side of relationship
fromtype[].*	<i>from</i> object	If type[] is specified, returns True if it is a type on the <i>from</i> side of relationship for this object; False if not. If no name is specified, returns names of types on the <i>from</i> side of the relationship.
hidden	Is this attribute defined as hidden?	True/False
id	Relationship identifier	Value
immediateattribute[].*	Non-inherited attributes	
immediatederivative[].*	The immediate child relationships	Returns first level children, see derivative for defaults.
immediatetrigger[].*	The triggers defined directly by the relationship excluding parent triggers	
kindof[].*	relationship.kindof[RELATIONSHIP_IN_HIERARCHY] can do the following: list all children of the specified parent relationship. Show the parent of the specified child relationship resolve a parent/child relationship to TRUE or FALSE	
modified	The time and date the administration object was modified	Value

modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	The attribute name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the relationship's name. If no string is given, the object type is relationship, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
preventduplicates	Prevent duplicate relationships?	True/False
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
toaction	?	Returns the name of the action rule (None , Float , or Replicate) on the <i>to</i> side of relationship
tocardinality	<i>to</i> object	Returns <i>to</i> cardinal ID
tocloneaction	<i>to</i> clone rule	Returns the name of the clone rule (None , Float , or Replicate) on the <i>to</i> side of relationship
tomeaning	<i>to</i> object	Returns <i>to</i> meaning
toreviseaction	<i>to</i> revision rule	Returns the name of the revision rule (None , Float , or Replicate) on the <i>to</i> side of relationship
totype[].*	<i>to</i> object	If type[] is specified, returns True if it is a type on the <i>to</i> side of the relationship for this object; False if not. If no name is specified, returns names of types on the <i>to</i> side of the relationship.
trigger[]	any trigger configured for the relationship	Returns list of triggers in this format: trigger = EVENT_INVOCATION:PROGRAM_NAME(ARGS) Include a program name in square brackets to check for the existence of that program: TRUE/FALSE will be returned to indicate if that program is or is not configured as a trigger. If you do not include a PROGRAM_NAME , all triggers are listed.
warnduplicates	Warning at attempt to create relationship with duplicate name?	True/False

Selectable Fields for Role

This section lists the selectable fields for roles.

Selectable Fields for Role		
Field	Description	Output
admin	?	Not enabled
assignment[].*	Person and role objects	Lists all assignments to persons and roles
ancestor[].*	Ancestor users	Without any " ." or [], returns a list of all ancestors of the current role. With [], returns TRUE/FALSE, depending on whether any ancestor matches the condition in []. If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string becomes the pattern.
child.*	Subrole object	Name of child role
description	?	Value
email	?	Value
hidden	Is this role defined as hidden?	True/False
id	The role identifier	Value
isagroup	Is a group?	True/False
isanorg	Is an organization?	True/False
isaperson	Is a person?	True/False
isaproject	Is a project?	True/False
isarole	Is a role?	True/False
mask	The list of accesses	Defaults to accesses of person
modified	The time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	The role name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the role's name. If no string is given, the object type is role, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
org[]'*	The immediate parent of the role defined as a project	Without any " ." or [], returns a list of all parents of the current role defined as an organization. With [],

		returns TRUE/FALSE, depending on whether any parent matches the condition in [].
		If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string becomes the pattern.
originated	The time and date the administration object was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
parent[].*	Parents of the role and group	Without any "." or [], returns a list of all parents. With [], returns TRUE/FALSE, depending on whether any parent matches the condition in []. If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string becomes the pattern.
person.*	The person(s) in the role	Name(s) of person(s) (see Selectable Fields for Person)
project[].*	The immediate parent of the role defined as a project	Without any "." or [], returns a list of all parents of the current role defined as a project. With [], returns TRUE/FALSE, depending on whether any parent matches the condition in []. If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string becomes the pattern.
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
role[].*	The immediate parent of the role	Without any "." or [], returns a list of all parents of the current role. With [], returns TRUE/FALSE, depending on whether any parent matches the condition in []. If the string in brackets has a bar character, the text before the bar is a pattern and the text after the bar is an expression that acts as a where-clause. If there is no bar, the entire string becomes the pattern.

Selectable Fields for Server

This section lists the selectable fields for server objects.

Selectable Fields for Server

Field	Description	Output
active	?	True/False
connect	?	Value
description	?	Value
hidden	Is this Store defined as hidden?	True/False
id	The store identifier	Value
modified	The time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Server Name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the server's name. If no string is given, the object type is server, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
timezone	?	Value
user	User used to access the database associated with the server.	Value

Selectable Fields for Store

This section lists the selectable fields for store objects.

Selectable Fields for Store		
Field	Description	Output
compressed	?	?
deletetrigger	The name of the java class that implements com.matrixone.fcs.mcs.DeleteTrigger interface.	Value
description	?	Value
encrypted	Not enabled	
hashed	Is this store hashed?	True/False
hidden	Is this Store defined as hidden?	True/False
host	The location of Store	Value
id	The store identifier	Value
location[].*	The alternate locations associated with captured stores.	Defaults to location names.
modified	The time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Store name	Value
numofversions	?	?
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the store's name. If no string is given, the object type is store, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
path	The path defined for the store.	Value
permission	The read/write permissions for owner group world	Value

property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
protocol	The protocol of captured store	One of ftp, ftps, file, or nfs.
replicationtrigger	The name of the java class that implements com.matrixone.fcs.mcs.ReplicationTrigger interface.	Value
search[].*	Used for URL-based full text search	If search[] is specified, returns information about Product?Line files matching query; otherwise, returns filenames.
size.[UNIT]	Used with hard link for FCS local clone feature (see <i>File Collaboration Server > Large Files > Hard Link for Local Clone</i>). Possible values for UNIT are: <ul style="list-style-type: none"> • byte • mb (megabyte) • gb (gigabyte) 	Total size of the files physically located in a store's path as computed by using ENOVIA V6 file metadata. Obsolete files also count.
type	The data type value	Returns type of store: captured.
url	The page from web server	Returns web page
prefix	The prefix defined for the store.	Value

Selectable Fields for Table

Selectables for properties related to geometry and display apply only to (Standard) User Tables.

Selectable Fields for Table		
Field	Description	Output
background +	The background color	Value
column[].*	The column names	Values
description	The table object description	Value
editorunits +	The unit chosen for the form's measurements	picas, points, or inches
footer +	The footer height in unit specified for table	Value
foreground +	The foreground color	Value
header +	The header height in unit specified for table	Value
height +	The height of table in unit specified for table	Value
hidden	Is this table defined as hidden?	True/False
id	The table identifier	Value
leftmargin +	The left margin width in unit specified for table	Value
modified	The time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	The table object name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the table's name. If no string is given, the object type is table, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	The time and date the administration object was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	The associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.

rightmargin +	The right margin width in unit specified for table	Value
width +	The width of table in unit specified for table	Value

+ Available only for (Standard) User Tables

Selectable Fields for Table Columns

For a list of selectables for columns in a table, enter `print table selectable column;` in MQL. Selectables for properties related to geometry and display apply only to User Tables. Selectables for properties related to URL links, settings, and user access apply to System Tables only.

Selectable Fields for Table Columns

Field	Description	Output
column.name	The column name	Value
column.description	The column description	Value
column.property[].*	The column properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
column.hidden	Is this column defined as hidden?	True/False
column.id	The column identifier	Value
column.user[].*	The user assigned to column	If user [] is specified, returns assigned user for column. If blank, returns assigned user for each column.
column.modified	The time and date the column was modified	Value
column.originated	The time and date the column was created	Value
column.type	The column type	Type assigned to column (always "Column")
column.number	The number assigned to column based on its creation order in table	Value
column.label	The column label	Value
column.value	The synonym for column label	Value
column.selectedvalue	[Unused]	
column.expression	Expression (select statement) associated with column	Type of selectable item (attribute, description, etc.)
column.expressiontype	Type of object expression applies to	Business object or relationship
column.font +	Font used to display column data	Value
column.editable	Is the column editable?	True/False
column.multiline	Can field display more than one line?	True/False
column.minwidth +	Minimum width of column	Value
column.minheight +	Minimum height of column	Value
column.autowidth	Does the column width autosize?	Standard Tables: True/False. System Tables: True
column.autoheight	Does the column height autosize?	Standard Tables: True/False. System Tables: True
column.absolutex +	Is horizontal location value absolute?	True/False
column.absolutey +	Is vertical location value absolute?	True/False
column.xlocation +	The horizontal location of field (left side)	Value
column.ylocation +	The vertical location of field (top)	Value
column.width +	The width of column in unit specified	Value

	for table	
column.height +	The height of column in unit specified for table	Value
column.href ‡	The column's href	URL plus parameters
column.alt ‡	The column's alt	Alt text
column.range ‡	JSP that gets a range of values and populates column with selected value.	JSP name
column.update ‡	The page to display after column is updated	URL
column.setting[].* ‡	Associated settings	If setting [] is specified, returns setting names and values. If blank, returns all setting names and values.

+ Available only for (Standard) User Tables ‡ Available only for System Tables

Selectable Fields for Type

This section lists the selectable fields for type objects.

Selectable Fields for Type

Field	Description	Output
abstract	Is this Type defined as abstract?	True/False
access	The category of access	Private/Protected/Public
application	Application owning this type	Application name
attribute[].*	Attributes of type	(see Selectable Fields for Attribute)
create[]	List of policies	If [policy_name] is specified returns true if the current user has create access in the first state of the named policy. If no policy is specified, returns a list of policies in which the current user has create access.
derivative[].*	All child types	If [name] is specified, returns True if name is child of this object; False if not. If no name is specified, returns names of all child types.
derived.*	Parent type	Name of the parent type definition
description	Type description	Value
fromrel[].*	'from' relationships, including those that specify a type from which the given type is derived	List of relationships that have the given type in their 'from' field
hidden	Is type defined as hidden?	True/False
id	Type identifier	Value
immediateattribute[].*	Non-inherited attributes	(see Selectable Fields for Attribute)
immediatederivative[].*	Immediate child types	returns first level children, see derivative for defaults.
method[].*	All methods that can be run on the type	List of methods for the specified type (see Selectable Fields for Program)
modified	Time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Type name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the type's name. If no string is given, the object type is type, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.

originated	Time and date the administration object was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
policy[].*	All policies governing the type	Value (see Selectable Fields for Policy)
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
torel[].*	A 'to' relationships, including those that specify a type from which the given type is derived	List of relationships that have the given type in their 'to' field

Selectable Fields for User

This section lists the selectable fields for user objects.

Selectable Fields for User		
Field	Description	Output
admin	List of administrative accesses	Defaults to administrative accesses of User.
description	?	Value
email	?	Value
hidden	Is this User defined as hidden?	True/False
id	User identifier	Value
isagroup	Is a group?	True/False
isaperson	Is a person?	True/False
isarole	Is a role?	True/False
mask	List of accesses	Defaults to accesses of User
modified	Time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	User name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the user's name. If no string is given, the object type is user, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
type	User type	Returns person, group, role, or association.

Selectable Fields for Vault

This section lists the selectable fields for vault objects.

Selectable Fields for Vault		
Field	Description	Output
description	?	Value
hidden	Is this Vault defined as hidden?	True/False
id	Vault identifier	Value
interface	Vault interface	Returns name of adaptlet .dll or shared library.
map	Map value for vault	Returns map file for adaplet.
maxoid	The highest value id that any of the vault's objects have	integer
modified	Time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Vault name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the vault's name. If no string is given, the object type is vault, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
server[].*	If vault is Foreign, returns server name.	If vault is local, returns nothing.
statisticsa	Total, max, min, and average number of object counts	Value
totalnumberofobjectsa	Count of all objects in vault	Value

a.) These values are computed dynamically by iterating over all objects in the vault. Therefore, they may be time-consuming to obtain.

Selectable Fields for Webreports

This section lists the selectable fields for webreports.

Selectable Fields for Webreport		
Field	Description	Output
name	Webreport name	Value
description	Not used	
property[].*	Associated properties	If property[] is specified, returns property name and values. If blank, returns all property names and values.
hidden	Is this webreport defined as hidden?	True/False
id	Webreport identifier	Value
modified	Time and date the webreport was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the webreport was modified	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the webreport's name. If no string is given, the object type is webreport, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Time and date the webreport was created	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the webreport was created	Value
searchcriteria	Expression string or object name	Value
notes	A string	Value
reporttype	A string	Value
groupby[].*	A string	If groupby[] is specified, returns groupby index and value. If blank, returns all groupby indices and values.
data[].*	A string	If data[] is specified, returns data index and value. If blank, returns all data indices and values.
user.*	The user's workspace to which the	Value

	webreport belongs	
visible[]	What users are visible?	List of users with visibility. If a user is specified, returns true or false.
appliesto	A string	One of "businessobjects", "relationships", or "both"
groupby[].maxgroupings	A integer	Value
##The following are valid for both <code>data</code> and <code>groupby</code> :		
groupby[]/data[].value	A string	The string expression for this groupby or data whether it was defined as a value string or by referring to an expression object.
groupby[]/data[].label	A string	The label for this groupby or data.
groupby[]/data[].expression	A string	Name of expression object, if any. (Empty string if defined without an expression object)
groupby[]/data[].appliesto	A string	One of "businessobjects", "relationships", or "both"
summary	A string	Value
##The following are valid for summary:		
summary.name	A string	Name of the summary.
summary.appliesto	A string	One of "businessobjects", "relationships", or "both"
summary.groupby[].*	A string	List of groupby indices in this summary
summary.data[].*	A string	List of data indices in this summary.
result	A boolean	True if there is a saved result; False if not.
archive	An integer	List of indices of the saved result archives.
##The following are valid for <code>result</code> and <code>archive</code> :		
result/archive[].created	A string	Returns date/time when created
result/archive[].creator	A string	Returns name of user who evaluated it
result/archive[].cell[].*	An integer	List of indices for all subsets defined by unique combinations of groupby values.
result/archive[].cell[].groupby	An integer	Number of groupby values defined for this cell. This will be the total number of groupbys for regular cells but will be a smaller number for summary cells.
result/archive[].cell[].groupby.index	An integer	For each cell this will give the list of indices for the groupbys defining the cell.
result/archive[].cell[].data	An integer	Number of data values defined for this cell. This will be the total number of datas for regular cells but will be a smaller number for summary cells.
result/archive[].cell[].data.index	An integer	For each cell this will give the list of indices for the datas defining the cell.
result/archive[].cell[].kindof	A string	One of "businessobject" or "relationship", indicating what the cell was evaluated against.
result/archive[].cell[].businessobject	An integer	Count of businessobjects associated with this cell.

result/archive[].cell[].relationship	An integer	Count of connections associated with this cell.
result/archive[].duration	A real	Returns how long (in seconds) it took to evaluate.
result/archive[].label	A string	Returns the result/archive label
result/archive[].description	A string	Returns the result/archive description
result/archive[].current	A boolean	Returns TRUE if webreport definition has not been changed since this result was computed, FALSE otherwise.
result/archive[].full	A boolean	Returns TRUE or FALSE depending on whether the computed data is complete. E.g., FALSE is returned if not all summaries or datas were computed.
result/archive[].value	A string	Returns the text that "evaluate webreport" would print if run with the same options as created these results
result/archive[] .xml	A string	Same as value except the results are printed out as an xml string.

Selectable Fields for Wizard

This section lists the selectable fields for wizards.

Selectable Fields for Wizard		
Field	Description	Output
code	Program code	For MQL or Tcl, lists the actual code; for external program, lists the command to launch the program.
description	Program description	Value
doesneedcontext	Does this program require a business object?	True/False
doesuseinterface	Is this program defined as launchable from a collaborative server application? (deferred and downloadable)	True/False
downloadable	Does program include code for operations not supported on the Web product?	True/False
execute	How is the execute flag set?	deferred or immediate
hidden	Is this program object hidden?	True/False
id	Program identifier	Value
isafunction	Is program used as a function? (trigger, state check/action, format edit/view/print, frame prologue/epilogue, widget load/validate, button command)	True/False
isamethod	Is this program associated with a type?	True/False
isjavaprogram	Is this program type Java	True/False
ismqlprogram	Is this program type MQL?	True/False
ispipedprogram	Is this program type piped	True/False
iswizardprogram	Is this program type a wizard	True/False
modified	Time and date the administration object was modified	Value
modified.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was modified	Value
name	Program name	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the wizard's name. If no string is given, the object type is wizard, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is

		returned.
originated	Time and date the administration object was originated	Value
originated.generic	The time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator) the administration object was created	Value
property[].*	Associated properties	If property[] is specified, returns property names and values. If blank, returns all property names and values.
type[].*	Includes all types which are derived from types specifying the program as a method (that is, all types for which the program is an inherited method)	Type for which the specified program is a method.

Selectable Fields for Workflow

This section lists the selectable fields for workflow objects.

Selectable Fields for Workflow		
Field	Description	Output
attribute[].*	Attribute instance	If attribute[name] is specified, then default is value; if not, default is a list of all associated attribute names.
automated.*	Automated activity	If automated[name] is specified, then default is value; if not, default is a list of all associated automated activity names.
automated.attribute[].*	Attributes of automated activity	Lists attributes of activity if nothing specified in []. With a value in [] returns attribute value. (see Selectable Fields for Attribute)
automated.description		Value
automated.history.*	Text log of history	Value
automated.lattice.*	Vault name	Vault name (see Selectable Fields for Vault)
automated.modified	Modification time	Value
automated.name	Automated activity name	Value
automated.originated	Creation time	Value
automated.owner.*	Person, role, or group	Owner name (see Selectable Fields for User)
automated.status	Automated activity status	Value
automated.trigger.*	Trigger program definitions	(see Selectable Fields for Program)
automated.type.*	Automated activity type	(see Selectable Fields for Process)
automated.vault.*	Vault name	(see Selectable Fields for Vault)
automated.xcoord	Horizontal location on graph	Value
automated.ycoord	Vertical location on graph	Value
automated.next	What comes after an automated activity	Value
automated.assignee.*	User assigned to an automated activity	Value
businessobject.*	Objects attached to the workflow	Business object name (see Selectable Fields for Business Objects)
description	Workflow description	Value
exists	Used with no other selectable; Asks: Does this object exist?	Returns true/false if it is the only selectable entered. Otherwise an error occurs.
history.*	Text log of history	Value
history.autosuspend	Workflow autosuspend user info and timestamp	Value
history.finish	Workflow finish user info and timestamp	Value
history.reassign	Workflow reassign user info and timestamp	Value
history.resume	Workflow resume user info and	Value

	timestamp	
history.start	Workflow start user info and timestamp	Value
history.stop	Workflow stop user info and timestamp	Value
history.suspend	Workflow suspend user info and timestamp	Value
id	Workflow identifier	Value
interactive[].*	Interactive activity	If interactive[name] is specified, then default is value; if not, default is a list of all associated interactive activity names.
interactive.attribute[].*	Attributes of interactive activity	Lists attributes of activity if nothing specified in []. With a value in [] returns attribute value (see Selectable Fields for Attribute)
interactive.description	?	Value
interactive.history.*	Text log of history	Value
interactive.lattice.*	Vault name	Value (see Selectable Fields for Vault)
interactive.modified	Modification time	Value
interactive.name	Interactive activity name	Value
interactive.originated	Creation time	Value
interactive.owner.*	Person, role, or group	Owner name (see Selectable Fields for User)
interactive.status	Interactive activity status	Value
interactive.trigger.*	Trigger program definitions	Value (see Selectable Fields for Program)
interactive.type.*	Interactive activity type	Value
interactive.vault.*	Vault name	Value
interactive.xcoord	Horizontal location on graph	Value
interactive.ycoord	Vertical location on graph	Value
interactive.next	What comes after an interactive activity	Value
interactive.assignee.*	User assigned to an interactive activity	Value
lattice.*	Vault name	Vault name (see Selectable Fields for Vault)
modified	Modification time and date	Value
originated.generic	Modification time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
name	Workflow name	Value
node[].*	Element of workflow	If node is specified, default is value. If nothing is specified in [], returns node names for workflow.
node.description	?	Node description
node.name	?	Node name
node.type	?	Node type
node.xcoord	Horizontal location on graph	Value
node.ycoord	Vertical location on graph	Value

node.next	Next process node	Value
object[]	a corresponding business object	Returns a business object where the type matches the string in [] and the name is the same as the workflow's name. If no string is given, the object type is workflow, with case ignored. In a case sensitive environment if more than 1 type meets that criteria (such as both a Person type and a PERSON type), OR if there are more than 1 object with the same type and name that are not in the same revision chain, the object with the latest originated date is returned. When objects with same type and name are in the same revision chain, the latest revision is returned.
originated	Creation time and date	Value
originated.generic	Creation time and date in generic format (independent of the MX_NORMAL_DATETIME_FORMAT and MX_DECIMAL_SYMBOL settings chosen by an Administrator)	Value
owner.*	Person, role, or group	Owner name (see Selectable Fields for User)
status	Workflow status	Value
type.*	Type object	Type name (see Selectable Fields for Type)
vault.*	Vault definition	Vault name (see Selectable Fields for Vault)

Macros

Macros provide a powerful tool for use in program objects. This section describes the available macros and how to use them.

In this section:

- [History and Common Usage](#)
- [Available Macros](#)
- [Macro Categories](#)
- [Trigger Macro Categories](#)
- [Trigger Event Macros](#)
- [Macro Processing and Command Syntax](#)

History and Common Usage

Macros are a simple name/value string substitution mechanism, where the macro value is substituted for the macro name when used in a Program. This single one-pass string substitution process occurs just prior to program execution. These macros are also created as a variable of the same name in the Runtime Programming Environment (RPE).

Since macros are also generated in the RPE, and because it provides greater flexibility, use of the RPE is strongly encouraged. Macros continue to be supported and provide a relatively easy way to work with variables. For example, a program used as the edit program in a Format would still likely use a macro to get the name of the checked in file to be edited.

The main limitation to macros is that they are available only for the program that calls them. When working with nested programs, the RPE enables you to pass values between programs, using `get env MACRO`.

Although the operating system determines the valid syntax of the commands, the typical syntax used is:

```
command ${ "MACRO 1" } ${ "MACRO 2" } ...
```

When using macros be sure to enclose them in quotes to allow spaces. (An exception to this rule is the OBJECT macro (see [Business Object Identification Macros](#)).

RPE variables can be redefined (and possibly unset) by subordinate programs, for example called programs, utility programs such as load and validate in wizards, trigger programs, range programs, and so on. To minimize the potential for unwanted name collisions, you should develop a naming convention for your RPE variables, and ALWAYS move macro variables into your RPE variables right away. For example, in method wizards, the first thing you should do in the opening frame's prologue program is to move TYPE, NAME, and REVISION into local wizard variables (typically something like `wizardname_TYPE`, `wizardname_NAME`, `wizardname_REV`).

Refer to the discussion of [Macro Processing and Command Syntax](#) for more information.

Also see the *ENOVIA Live Collaboration Installation Guide* for descriptions of the environment variables `MX_JIT_TRIGGER_MACROS` and `MX_NESTED_TRIGGER_MACROS`.

Macros are populated in the RPE as each subordinate program is called. These macros will remain in the RPE until the parent program execution has completed. With `MX_JIT_TRIGGER_MACROS=true`, users should be aware that there are conditions where these macros will not be able to be evaluated outside the subordinate program for which they were instantiated--such as a disconnect trigger that removes a connection between 2 businessObjects. In this example, the macro `ISFROZEN` will be available, but if a user attempts to get this value outside the disconnect trigger, they will receive an error.

Available Macros

This section lists the available macros.

In ENOVIA Live Collaboration, programs can be used as a:

- Standalone program or Tool
- Method
- View, edit, or print program in a Format
- Lifecycle check or action
- Trigger check, override, or action program
- Attribute's program range
- Wizard load, validate, prologue, epilogue, or button program
- Workflow AutoActivity

Different macros are populated with values depending on how the program is used. The table below shows each possible use of a Program, and the category of macros available to it. Macro category tables are provided [Macro Categories](#).

If a program is used as:	the available macros are:
a standalone program or Tool (Includes Wizards)	Intrinsic Macros
a Method (Includes Wizards)	Intrinsic Macros
	Business Object Identification Macros
a view, edit, or print program in a Format	Intrinsic Macros
	Format Macros
a lifecycle check or action	Intrinsic Macros
	Lifecycle Check and Action Macros
a Trigger check, override, or action program	Intrinsic Macros
	The Standard Macros table in Trigger Macro Categories
	Depends on Event, See Trigger Event Macros
an attribute's program range	Intrinsic Macros
	Program Range Macros
a Wizard load, validate, prologue, epilogue, or button program.	Intrinsic Macros
	Wizard Variables
a Workflow AutoActivity.	Intrinsic Macros
	the AutoActivity Event macros table in Activity Event Macros

Macro Categories

The specific macros that can be used depend on the objects or functions being handled or implemented as defined in this section.

In this section:

- [Intrinsic Macros](#)
- [Business Object Identification Macros](#)
- [Format Macros](#)
- [Lifecycle Check and Action Macros](#)
- [Wizard Variables](#)
- [Program Range Macros](#)
- [Dynamic UI Macros](#)
- [Connection Macros](#)
- [Workflow Macros](#)
- [JPO Macros](#)

Intrinsic Macros

The table below lists macros that are available to all programs regardless of how it is invoked:

Intrinsic Macros	
Macro	Meaning
USER	User of the current context. When used within a shell command, USER is the user ID from the operating system, which is not necessarily the same as the ENOVIA session user.
VAULT	Vault of the current context. Not available on relationship triggers since relationships can connect objects in 2 different vaults. (But is available on to/from/connect/disconnect events on business objects.)
INVOCATION	Tells how the program was invoked. Can have the following values: <code>method</code> , <code>program</code> , <code>format</code> , <code>check</code> , <code>action</code> , <code>override</code> , <code>range</code> , <code>load</code> , <code>validate</code> , <code>prologue</code> , <code>epilogue</code> , <code>button</code> , <code>autoactivity</code> , <code>observer</code> .
ARGCOUNT	Number of program parameters
TRANSACTION	Where the program was executed in terms of transaction boundaries. Potential values are <code>read</code> , <code>update</code> or an empty string, which means that it was executed outside of transaction boundaries.
HOST	Name of host machine that launched the current program
APPLICATION	Name of the application that launched the current program. Possible values are <code>MQL</code> , <code>matrix</code> , or <code>BOS</code> .
LANGUAGE	Language as defined in the initialization file in the variable <code>MX_LANGUAGE_PREFERENCE</code> .
HOME	User's home directory
PATH	Path statement of the operating system
MATRIXHOME	Directory that contains the connection file
MATRIXPATH	Path to the executable file
SELECTEDOBJECTS	<p>List of currently selected objects to be passed to any program. This macro is populated whenever a program or method is invoked from the toolbar, tool menu (Web Navigator only), right-mouse menu or Methods dialog. The macro is also populated when certain dialogs are launched via the <code>appl</code> command (appl icons, details, indented, star and indentedtable), as long as a dismiss program is defined. The macro's value consists of a single string of space-delimited business object IDs for the objects that are selected at the time the program or method is invoked. (For a method, this macro is redundant--it is equivalent to the OBJECTID macro--but it is populated nevertheless for consistency.)</p> <p>The SELECTEDOBJECTS macro can be read into a Tcl string or list variable as follows:</p> <pre># this reads the macro as a single Tcl string set sObjs [mql get env SELECTEDOBJECTS] # this reads the macro into a Tcl list. set env lObjs [split [mql get env SELECTEDOBJECTS]]</pre> <p>When no objects are selected, the macro is created, but holds an empty string.</p>
WORKSPACEPATH	Directory that is used as the base directory for upload/download. For programs/wizards run through the server, it is evaluated by combining the enovia.ini settings <code>MX_BOS_ROOT</code> and <code>MX_BOS_WORKSPACE</code> , and appending a unique temporary directory for the session that runs it to avoid collisions across sessions. For desktop client configurations, it is set to the enovia.ini setting <code>TMPDIR</code> . These settings guarantee that the client has access to the files on the server.

Business Object Identification Macros

Most programs that use a business object have access to the following.

Business Object Identification Macros

Macro	Meaning
TYPE	Business object type
NAME	Business object name
REVISION	Business object revision
OBJECT	Full business object specification (Type, Name, Revision). Do not use double quotes for this macro, since it evaluates to more than a single item and each item (Type, Name, and Rev) in the output will be enclosed in double quotes.
OBJECTID	The system generated numerical identification of the object

Format Macros

Programs that are used for file access can use Format macros.

Format Macros	
Macro	Meaning
FILENAME	Name of the first file in the default format or the selected file. During a checkout operation, includes the full target directory/filename. (For desktop applications, this is the final destination for the file. For Studio Customization Toolkit programs, this is the workspace directory on the server, prior to copying to the client.)
FILENAMES	Names of all the files in the default format, space delimited
FILENAME_ORIGINAL	Name of the first file in the default format or the selected file, as it is recorded in the database (hash name if using a hashed captured store), and also as it will be named on the client system after an Studio Customization Toolkit checkout operation.
FORMAT	Default format of the business object or the currently selected format
PROG	PROG is the full command line (with or without arguments) returned by Windows file associations.

Lifecycle Check and Action Macros

Programs used for lifecycle checks and actions can use the macros shown below.

Lifecycle Check and Action Macros

Macro	Meaning
EVENT	Most recent transaction performed. For lifecycle checks or actions, EVENT can be <code>create</code> , <code>revisioned</code> , <code>copy</code> , <code>promote</code> , <code>demote</code> , or <code>change policy</code> .
OBJECT	Full specification of object (TYPE, NAME, and REVISION)
TYPE	Object's type
NAME	Object's name
REVISION	Object's revision

Wizard Variables

Wizards can use the following RPE variables. RPE variables are accessed with 'get env ...', while MACROs can be used with \${} or as an RPE variable.

Wizard Variables

Macro	Meaning
WIZARDNAME	Name of the Business Wizard.
FRAMENAME	Name of the current frame.
FRAMENUMBER	Number of the current frame in the frame sequence (excluding master and status frames). Frame numbers begin with 1. The status frame returns a value of 0.
FRAMETOTAL	Total number of frames in the frame sequence (excluding master and status frames).
FRAMEMOTION	Current state of wizard processing: start The wizard has been started; the user has not yet clicked a button. back The user clicked the Back button. next The user clicked the Next button. finish The user clicked the Finish button. repeat The current frame has been redisplayed. status The current frame is the status frame of the wizard.
WIZARDARG	A single RPE variable that is created from the ARG string in an <code>appl wizard</code> command, which can be read by any of the wizard's supporting code to receive input from its 'caller.' If you want to pass multiple pieces of information from the caller to the wizard, you would need to format the information into a single string with an identifiable delimiter (for example,). For example, to pass a businessobject type, current state and owner, you could collect them into a Tcl variable sArgString as follows: <pre>set sArgString "\$sType \$sState \$sOwner" appl wizard bus \$sBusId MyWizard \$sArgString</pre> Then, the wizard program that reads the WIZARDARG RPE variable could split it up again as follows: <pre>set sArgString [mql get env WIZARDARG] set lArgList [split \$sArgString] set sType [lindex \$lArgList 0] set sState [lindex \$lArgList 1] set sOwner [lindex \$lArgList 2]</pre>

Program Range Macros

Attribute range programs can use the following macros.

Program Range Macros	
Macro	Meaning
OBJECT	Owning Information for Objects. Note that OBJECTID is NOT available.
TYPE	
NAME	
REVISION	
RELID	Owning Information for Relationships
EVENT	EVENT will be "attribute choices" or "attribute check". When EVENT equals "attribute check", the ATTRVALUE macro will also be provided.
ATTRVALUE	The value of the attribute as set for the specified business Object. This macro is only provided when EVENT equals "attribute check"
ATTRNAME	The attribute name.
ATTRTYPE	The type of attribute. Value can be "String," "Real," "Integer", "Date," "Boolean."

Dynamic UI Macros

The following tables provide lists of macros used in the configuration parameters of the administrative menu and command objects. These menu and command objects are used for configuring the Menus/Trees/Actionbars in the ENOVIA products that use them. These are the only macros currently supported for use in any dynamic UI component.

The following topics are discussed:

- [Directory Macros](#)
- [Object Property Macros](#)

Directory Macros

The following table provides the list of directory specific macros used in the configuration setting.

Directory Macros

Macro Name	Description
COMMON_DIR	To substitute the "common" directory below "ematrix" directory. The substitution is done with reference to any application-specific directory and it is relative to the current directory.
ROOT_DIR	To substitute the "ematrix" directory. The substitution is done with reference to any application-specific directory below "ematrix" and it is relative to the current directory.
SUITE_DIR	The macro to substitute the application-specific directory below "ematrix" directory. The substitution is done based on the "Suite" to which the command belongs and it is relative to the current directory.



Object Property Macros

The following table provides the list of Object specific macros supported by the dynamic UI components.

Object Property Macros

Macro	Meaning
\${TYPE}	This will substitute the type name of the current business object. This Macro is currently used only in configuration of the Label for Root tree node. The tree type must be associated with a business object and require a valid business object id to process the macro substitution.
\${NAME}	This will substitute the name of the current business object. This Macro is currently used only in configuration of the Label for Root tree node. The tree type must be associated with a business object and require a valid business object id to process the macro substitution.
\${REVISION}	This will substitute the type name of the current business object. This Macro is currently used only in configuration of the Label for Root tree node. The tree type must be associated with a business object and require a valid business object id to process the macro substitution.

Connection Macros

Program objects that are executed as methods have two macros available when launched from a Navigator window.

Connection Macros

Macro	Meaning
CONNECTIONID	The relationship ID of the connection next to the selected child business object in the Navigator browser.
CONNECTIONDIRECTION	The relationship direction (<i>to</i> or <i>from</i>) of the connection next to the selected child business object in the Navigator browser.

The connection macros are available only when the program is launched as a method from an object displayed in a Navigator window AND if the object is not the expanded object. Both macros will be empty if:

- no connection is displayed, as in the case when the selected business object is in a flat browser;
- more than one connection is shown, as when the selected object is in the center of a star browser or at the top of an indented browser.

In some cases the macros will not even be placed in the RPE. This situation occurs when a method is invoked programmatically (either from a program object or interactive MQL) or if a program is not executed as a method (i.e. from the toolbar).

Most programs that use a connection have access to the following:..

Connection Identification Macros

Macro	Meaning
RELID	The system generated numerical identification of the connection
ISFROZEN	Boolean

Similarly, when used in the context of a connection, programs can access the following macros to identify objects on the ends of the relationship:

From Connection Macros

Macro	Meaning
FROMOBJECTID	Object Identifier of Connection on "from" end of connection
FROMISFROZEN	
FROMISRELATIONSHIP	

To Connection Macros

Macro	Meaning
TOOBJECTID	Object Identifiers of Connection on "to" end of connection
TOISFROZEN	
TOISRELATIONSHIP	

Workflow Macros

Workflows can contain 4 types of programs: check, override, and action event triggers, as well as Autoactivity programs.

The Macros available for Workflow trigger programs are listed in the [Process Event Macros](#) table. The macros below are available to Autoactivity programs.

Auto Activity Macros

Macro	Meaning
PROCESS	These are Process Macros (see Trigger Macro Categories .)
WORKFLOW	
WORKFLOWOWNER	
WORKFLOWID	
AUTOACTIVITYTYPE	Name of automated activity definition defined in Business Modeler
AUTOASSIGNEE	Name of automated activity assignee
AUTOPROGRAMNAME	Name of automated activity program
INPUTARGS	Input arguments for the program object
AUTOACTIVITY	Name of automated activity instance
AUTOACTIVITYID	System generated activity ID
AUTOACTIVITYOWNER	Name of activity owner

JPO Macros

Java Program Objects have two macros available.

JPO Macros

Macro	Meaning
<code>CLASSNAME</code>	Refers to the JPO name in the Java class definition.
<code>CLASS:jpo_name</code>	Refers to other JPO classes inside the code of a JPO, for example when extending a base class or newing up an object. <code>jpo_name</code> is the name of the JPO holding the class of interest

There are two problems with using schema object names for the JPO class. The first problem is that schema object names are allowed to contain spaces and other non-alpha characters that would cause errors in the Java compiler. The second problem is dynamic changes to the schema object names would not be automatically made to the code within JPOs.

Special macros are used to map the schema object names to legal Java-compliant names (a process known as *name mangling*). These macros are also appropriately modified automatically when a schema object name is changed.

Use of these parameterized class names allows both dynamic loading of classes after the source has been changed without restarting the system and automatic generation of dependencies between multiple Java Program Objects.

For example, if there is JPO named "RADProject" and its corresponding Business Type RADProject derives from Project, then the Java code defining the class for RADProject would look something like:

```
public class ${CLASSNAME} extends ${CLASS:Project}{:}
```

During compilation, the macro `${CLASSNAME}` is mangled into a name that includes the JPO name "RADProject" (plus a suffix of "_mxJPO" followed by a hashed encoding). The macro `${CLASS:Project}` is mangled into the exact same name as it was for `${CLASSNAME}` appearing in the Project JPO. In other words, the system does all the appropriate name mangling and the JPO programmer needs to be concerned only with the higher-level macros.

For example, the "Hello World" JPO ends up with a class name something like "Hello_World_mxJPOTaxMTwEAAAFAAAA" (the hashed encoding at the end will vary).

Trigger Macro Categories

The following tables list macros in categories. These categories are referenced later for each trigger supported event. Standard macros are available to programs used for all trigger events.

Trigger programs have access to [Intrinsic Macros](#), as well as others that are applicable to the specific event.

Macro (and RPE variable) values remain the same between activation of the check program, the override program, and the action program. You need to take care when interpreting the meaning of macros. For example, there are macros called `OWNER` and `NEWOWNER` for the changeowner event, which makes clear sense for the check program and override program, but by the time the action program executes, the owner may have changed. No additional macro called `OLDDOWNER` is necessary, but the action program must use `OWNER` to mean "old owner" and `NEWOWNER` to mean "current owner."

Standard Macros

Macros	Meaning
TIMESTAMP	The current date/time
EVENT	Most recent transaction performed. <code>EVENT</code> for triggers will equal any of the Supported Trigger Events .

Context/Access Macros

Macro	Meaning
USER	Current user
ACCESSFLAG	Access flag (<code>True</code> if <code>USER</code> will be allowed to perform event)
CHECKACCESSFLAG	Check access flag (<code>False</code> when access is not checked, such as during import)
ACCESS	The current access being checked
RULEUSER	The user for which a rule is defined
CHECKEDUSER	The current user being checked (This macro can only be evaluated in the context of an access filter.)
CHECKEDUSER_PRJ	The projects of <code>CHECKEDUSER</code>
CHECKEDUSER_ORG	The organizations of <code>CHECKEDUSER</code>
CHECKEDUSER_ROLE	The roles of <code>CHECKEDUSER</code>

Business Object Macros

Macro	Meaning
Business Object Identification Macros	Object Identifiers
OWNER	Object Settings and descriptions
DESCRIPTION	
CURRENTSTATE	
ENFORCEDLOCKINGFLAG	
LOCKFLAG	
LOCKER	
POLICY	

State Macros

Macro	Meaning
ISENABLED	Boolean State Flags

ISDISABLED
ISOVERRIDDEN
ISCURRENT
ISREVISIONABLE
ISVERSIONABLE
HASSCHEDULEDDATE
HASACTUALDATE
AUTOPROMOTE

Signature Macros

Macro	Meaning
SIGNATURE	Current Signature Name
ISSIGNED	Boolean Signature Flags
ISAPPROVED	
ISREJECTED	
ISIGNORED	
EXISTINGSIGNER	If signed, the signer's Name.
EXISTINGCOMMENT	Associated comment.

Relationship Macros

Macro	Meaning
RELTYPE	Relationship Name
RELID	Relationship ID, available for all relationship trigger events.
FROMMEANING	From side meaning
FROMTYPES	From business Types allowed (excluding child Types)
FROMALLFLAG	Boolean flag - <code>True</code> if all business Types allowed
FROMREVACTION	Revision action of Object at From end. Value= <code>Float</code> , <code>Replicate</code> , or <code>None</code> .
FROMCARD	Cardinality of From connection. Value= <code>One</code> or <code>N</code>
TOMEANING	To side meaning
TOTYPES	To business Types allowed (excluding child Types)
TOALLFLAG	Boolean flag - <code>True</code> if all business Types allowed
TOREVACTION	Revision action of Object at To end. Value= <code>Float</code> , <code>Replicate</code> , or <code>None</code> .
TOCARD	Cardinality of To connection. Value= <code>One</code> or <code>N</code>

Attribute Macros

Macro	Meaning
ATTRNAME	Name of attribute
ATTRVALUE	Current value
ATTRRTYPE	Type (<code>Integer</code> , <code>Real</code> , <code>String</code> , <code>Boolean</code> , or <code>Date</code>)

ATTRDEFAULT	Default value
ATTRCHOICES	List of value choices

Process Macros

Macro	Meaning
PROCESS	Name of the process definition defined in Business Modeler
WORKFLOW	Name of workflow defined in Matrix Navigator
WORKFLOWOWNER	Name of workflow owner defined in Matrix Navigator
WORKFLOWID	System generated workflow ID of this workflow instance. This ID can be used in place of processname workflowname in commands like print workflow or modify workflow. It is NOT a businessobject ID.

Interactive Activity Macros

Macro	Meaning
ACTIVITYTYPE	Name of activity definition defined in Business Modeler
ASSIGNEE	Name of user that process has assigned to the activity in Business Modeler
ACTIVITY	Name of activity instance created in ENOVIA Live Collaboration
ACTIVITYID	System generated activity ID
ACTIVITYOWNER	Name of activity owner

Auto Activity Macros

Macro	Meaning
AUTOACTIVITYTYPE	Name of automated activity definition defined in Business Modeler
AUTOASSIGNEE	Name of automated activity assignee
AUTOPROGRAMNAME	Name of automated activity program
INPUTARGS	Input arguments for the program object
AUTOACTIVITY	Name of automated activity instance
AUTOACTIVITYID	System generated activity ID
AUTOACTIVITYOWNER	Name of activity owner

Query Variables/Macros

Variable	Meaning
TYPE_PATTERN*	The value entered in the Type field of the Find or Find Like windows, or the TYPE_PATTERN specified in MQL/Studio Customization Toolkit programs.
NAME_PATTERN*	The value entered in the Name field of the Find or Find Like windows, or the NAME_PATTERN specified in MQL/Studio Customization Toolkit programs.
REVISION_PATTERN*	The value entered in the Revision field of the Find or Find Like windows, or the REVISION_PATTERN specified in MQL/Studio Customization Toolkit programs.
VAULT_PATTERN*	The value entered in the Vault field of the Find or Find Like windows, or the VAULT_PATTERN specified in MQL/Studio Customization Toolkit programs.
OWNER_PATTERN	The value entered in the Owner field of the Find or Find like windows, or the OWNER_PATTERN specified in MQL/Studio Customization Toolkit programs.
WHERE_PATTERN	The value entered in the Where field of the Find or Find like windows, or the WHERE_PATTERN specified in MQL/Studio Customization Toolkit programs.
EXPANDTYPE	Boolean value specified in the query.

LIMIT	Integer value specified in the query. May be set as environment variable. The setting in the query trigger will overwrite the setting in the environment.
-------	---

When resetting these as RPE variables in a ValidateQuery program, you must make them global.

Trigger Event Macros

This section lists the macros listed for each event. Trigger programs also have access to intrinsic and standard macros.

In this section:

- [Attribute Event Macros](#)
- [Business Object Event Macros](#)
- [Connection Event Macros](#)
- [State Event Macros](#)
- [Process Event Macros](#)
- [Activity Event Macros](#)
- [Query Event Macros](#)

Attribute Event Macros

Attributes can be assigned a Modify Trigger which has access to the following macros.

Attribute Event Macros		
Event	Macros Available	Comments
modify	See Attribute Macros in Trigger Macro Categories	
	ATTRDESCRIPTION	Description of the attribute as defined by Business Administrator.
	USER	Logged in user of session context.

Modify event triggers on an attribute are executed when that particular Attribute of a business object or connection is modified. However, if the Type or Relationship has any `modifyattribute` event triggers defined they are also launched when this or *any* attribute of the connection or business object is modified.

The `modifyattribute` event has many more macros available to it. Refer to the modify attribute event listing in [Business Object Identification Macros](#) for more information.

Business Object Event Macros

The table below shows the specific macros available for the trigger events which are related to Object Types.

Business Object Event Macros		
Event	Macros Available	Comments
changename	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	NEWNAME	target name
	NEWREV	target revision
changeowner	ALTOWNER1	Old altowner1 (ALTOWNER1 is an optional user)
	ALTOWNER2	Old altowner2 (ALTOWNER2 is an optional user)
	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	KINDOFOWNER	Indicates the type of owner that was set. Possible values are owner, altowner1 or altowner2.
	NEWOWNER	Target owner
	NEWALTOWNER1	Target altowner1
	NEWALTOWNER2	Target altowner2
changepolicy	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	NEWPOLICY	Target policy
changetype	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	NEWTYPE	Target type
changevault	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	NEWVAULT	target vault
checkin	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	FORMAT	target format (if blank, default format used)
	FILENAME	target file- Note that FILENAMES is not available since for each file the user selects, a separate checkin event is performed.

	APPENDFLAG	append flag (<code>True</code> = append, <code>False</code> = overwrite)
checkout	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	FORMAT	target format (if blank, default format used)
	FILENAME	target file (if blank checkout all files) Note that FILENAMES is not available since for each file the user selects, a separate checkout event is performed.
connect	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	Relationship Macros in Trigger Macro Categories	
	CONNECTION	end of relationship (<code>From</code> or <code>To</code>)
copy (clone)	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	NEWNAME	
	NEWREV	
	NEWVAULT	Target vault (if blank, source vault used)
create	<i>Unlike all other Business Object events, the "create" event has no Business Object on which to operate.</i>	
	USER	These are 2 of the 3 Context/ Access Macros in Trigger Macro Categories . ACCESSFLAG is not available.
	CHECKACCESSFLAG	
	OBJECT	New Object's Identifiers.
	TYPE	
	NAME	
	REVISION	
	OBJECTID	This Macro is only available to the Action trigger program, after the object has actually been created.
	POLICY	Target Policy
	TRIGGER_VAULT	Target Vault
delete	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	MX_OBJECTS_BEING_DELETED	A list of ids of objects in the process of being deleted, separated by the " " character.
disconnect	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	Relationship Macros in Trigger Macro Categories	
	CONNECTION	end of relationship (<code>From</code> or <code>To</code>)

grant	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	GRANTEE	A comma separated list of grantees to which the object is being granted.
	GRANTOR	Person granting the object, basically the current context.
	GRANTEEACCESS	List of accesses being granted to the grantees.
revoke	GRANTEE SIGNATURE	Boolean indicating whether or not the grantee can sign a signature for the grantor.
	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	GRANTEE	The grantee that is being revoked.
lock	GRANTOR	The grantor that is being revoked. It is not required to be the current context.
	Business Object Macros in Trigger Macro Categories	
modifyattribute	Context/ Access Macros in Trigger Macro Categories	
	Attribute Macros in Trigger Macro Categories	
	NEWATTRVALUE	target attribute value
	CHECKRANGEFLAG	Boolean check range flag - <code>FALSE</code> when range checking is not performed such as during import.
	NEWDESCRIPTION	target description
removefile	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	FORMAT	target format (if blank, default format used)
	FILENAME	target file (if blank remove all files) Note that FILENAMES is not available since for each file the user selects, a separate <code>removefile</code> event is performed.
	NEWREV	target revision
unlock	Business Object Macros in Trigger Macro Categories	

Context/ Access Macros in
[Trigger Macro Categories](#)

Connection Event Macros

The table below shows the available macros for each supported connection event.

Connection Event Macros		
Event	Macros Available	Comments
create event	Relationship Macros in Trigger Macro Categories Context/ Access Macros in Trigger Macro Categories FromBusinessObject Macros in Business Object Identification Macros ToBusinessObject Macros in Business Object Identification Macros	
delete event	Relationship Macros in Trigger Macro Categories Context/ Access Macros in Trigger Macro Categories FromBusinessObject Macros in Business Object Identification Macros ToBusinessObject Macros in Business Object Identification Macros	
freeze event	Relationship Macros in Trigger Macro Categories Context/ Access Macros in Trigger Macro Categories FromBusinessObject Macros in Business Object Identification Macros ToBusinessObject Macros in Business Object Identification Macros	
modifyattribute event	Relationship Macros in Trigger Macro Categories Context/ Access Macros in Trigger Macro Categories Attribute Macros in Trigger Macro Categories NEWATTRVALUE CHECKRANGEFLAG FromBusinessObject Macros in Business Object Identification Macros ToBusinessObject Macros in Business Object Identification Macros	target attribute value check range flag
thaw event	Relationship Macros in Trigger Macro Categories Context/ Access Macros in Trigger Macro Categories FromBusinessObject Macros in Business Object Identification Macros ToBusinessObject Macros in Business Object Identification Macros	

State Event Macros

The following table shows the macros that can be used in lifecycle event triggers.

State Event Macros		
Event	Macros Available	Comments
approve event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	Signature Macros in Trigger Macro Categories	
	STATENAME	target state
demote event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	STATENAME	source state
	NEXTSTATE	target state, if demotion succeeds
disable event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	STATENAME	target state
enable event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	STATENAME	target state
ignore event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	Signature Macros in Trigger Macro Categories	
	STATENAME	target state

	COMMENT	associated comments
override event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	STATENAME	target state
promote event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	STATENAME	source state
	NEXTSTATE	target state, if promotion succeeds
reject event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	Signature Macros in Trigger Macro Categories	
	STATENAME	target state
	COMMENT	associated comments
schedule event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	STATENAME	target state
	SCHEDULED	scheduled date and time
unsign event	Business Object Macros in Trigger Macro Categories	
	Context/ Access Macros in Trigger Macro Categories	
	State Macros in Trigger Macro Categories	
	Signature Macros in Trigger Macro Categories	
	STATENAME	target state

Process Event Macros

The following macros are available to Process trigger programs.

Process Event Macros		
Event	Macros Available	Comments
FinishProcess	Process Macros in Trigger Macro Categories	
ReassignProcess	Process Macros in Trigger Macro Categories	
ResumeProcess	Process Macros in Trigger Macro Categories	
StartProcess	Process Macros in Trigger Macro Categories	
StopProcess	Process Macros in Trigger Macro Categories	
SuspendProcess	Process Macros in Trigger Macro Categories	

Activity Event Macros

Macros available to Interactive and Automatic Activity triggers are shown in the tables that follow.

Interactive Activity Event Macros

Event	Macros Available	Comments
ActivateActivity	Process Macros in Trigger Macro Categories	
	Interactive Activity Macros in Trigger Macro Categories	
CompleteActivity	Process Macros in Trigger Macro Categories	
	Interactive Activity Macros in Trigger Macro Categories	
OverrideActivity	Process Macros in Trigger Macro Categories	
	Interactive Activity Macros in Trigger Macro Categories	
ReassignActivity	Process Macros in Trigger Macro Categories	
	Interactive Activity Macros in Trigger Macro Categories	
ResumeActivity	Process Macros in Trigger Macro Categories	
	Interactive Activity Macros in Trigger Macro Categories	
SuspendActivity	Process Macros in Trigger Macro Categories	
	Interactive Activity Macros in Trigger Macro Categories	

AutoActivity Event Macros

Event	Macros Available	Comments
ActivateAutoActivity	Process Macros in Trigger Macro Categories	
	Auto Activity Macros in Trigger Macro Categories	
CompleteAutoActivity	Process Macros in Trigger Macro Categories	
	Auto Activity Macros in Trigger Macro Categories	
OverrideAutoActivity	Process Macros in Trigger Macro Categories	
	Auto Activity Macros in Trigger Macro Categories	
ReassignAutoActivity	Process Macros in Trigger Macro Categories	
	Auto Activity Macros in Trigger Macro Categories	

ResumeAutoActivity	<p>Process Macros in Trigger Macro Categories</p> <p>Auto Activity Macros in Trigger Macro Categories</p>
SuspendAutoActivity	<p>Process Macros in Trigger Macro Categories</p> <p>Auto Activity Macros in Trigger Macro Categories</p>

Query Event Macros

Query triggers are quite different than other triggers.

Refer to [Validate Query Trigger](#) for more information.

Query Event Macros

Event	Macros Available	Comments
Query	Intrinsic Macros Standard Macros in Trigger Macro Categories Query Variables/Macros in Trigger Macro Categories	

If the ValidateQuery program is a JPO, no macros are available as input arguments. To work around this issue, the ValidateQuery program should be a Tcl program that calls the JPO to do the actual work.

Macro Processing and Command Syntax

"Macro processing" involves the replacement of macro names with their current values to create the precise code that is executed from a Program object or with the `shell` command.

Macro values are not evaluated unless and until they are actually used. For example, a modify attribute trigger will not cause an attribute range program to be executed unless the macro `${ATTRCHOICES}` is explicitly used by the program.

Macros can affect the surrounding text if the required characters have additional uses in programming, such as "\$" or "\". To include a literal "\${" in your text, you must turn off the macro processing substitution on a case-by-case basis by using the escape character "\".

For example, if `USER=guest` then:

```
output "The value of \${USER} is ${USER}"
```

results in the output:

```
The value of ${USER} is guest
```

You may want the "\$" in the text to be macro processed. This can be accomplished by using a pair of backslashes. For example, if `MATRIXHOME=tools\matrix-10.7\` then:

```
output "The file can be found in \\${MATRIXHOME}"
```

results in the output:

```
The file can be found in \tools\matrix-10.7\
```

So, to restate, macro processing of a backslash (\):

- followed by a "\$", results in a literal \$
- followed by another backslash, results in a single backslash
- in all other cases is treated literally

The consequence of this means that any backslashes that need to appear in pairs (like the beginning of a UNC path) will require escaping twice by using four backslashes. For example, one might use:

```
shell "\\\\\\ourserver\Matrix107\bin\winnt\mql.exe -v";
```

Note that macro processing is done in a single pass. Therefore, the values substituted in for macro names are not subject to macro processing. For example, if: `MATRIXHOME=\matrix107`, then one could use:

```
shell "${MATRIXHOME}\bin\winnt\mql.exe -v";
```

Also note that the Tcl language also recognizes the backslash (among other characters) as an escape character. This means any Tcl code in a Program may require additional escaping. Refer to Tcl documentation for information on escaping in Tcl.

Also, the escape keyword can be used as the first word in any where clause to cause EENOVIA Live Collaboration to allow backslash (\) to be treated as an escape character; that is, to remove all special meaning from the character after the backslash.

Refer to the "Working with Workspace Objects" chapter in the *MQL Guide*, for more information and examples with the escape character.

When a program file name specified in an add or modify command includes the _mxJPO suffix, such as `test_mxJPO.java`, the system behaves the same as it does with the insert command -- that is, it does the macro processing and then inserts the code. For program files that do not include the _mxJPO suffix as part of the filename, the system does no macro processing on the code. Therefore, it is recommended that the _mxJPO suffix is routinely used.

.NET Support for the Studio Customization Toolkit

A .NET assembly of eMatrixClientXML.dll provides ENOVIA Live Collaboration integration with any .NET language, such as C#, VB.NET, J#, and Managed C++. eMatrixClientXML.dll exposes classes found in eMatrixAppletXML.jar, enabling .NET developers to reference the DB and Util Java class packages. For information about these packages, see the Java Docs installed with the Studio Customization Toolkit.

In this section:

- [eMatrixClientXML.dll](#)
- [.Net Sample Application](#)
- [.NET Limitations](#)

eMatrixClientXML.dll

You need to consider the issues in this section when using eMatrixClientXML.dll in a .NET environment.

- The eMatrixClientXML.dll can be used either as a private or shared assembly. If shared, be sure to register eMatrixClientXML.dll in the system Global Assembly Cache (GAC).
- The eMatrixClientXML.dll .NET clients will need to install the following .NET components, which can be downloaded from Microsoft at <http://www.microsoft.com/downloads/en/default.aspx>:
 - .NET framework runtime
 - J# redistributable package
- Developers using .NET languages other than J# must add a reference to the Microsoft Visual J# .NET Class Library (vjslib) in order to integrate with eMatrixClientXML.dll. This is usually done in the References section of your .NET language developer's tool.

.Net Sample Application

The VB.NET module in this section illustrates how to open a business object and print its attributes. It imports the matrix.db package to implement business object methods. It also imports the matrix.util package to use an exception method.

In this section:

-  [Sample Code](#)
-  [Building a Sample Application](#)

Sample Code

This section provides sample code to open a business object and print its attributes.

```
Imports System
Imports Microsoft.VisualBasic
Imports matrix.db
Imports matrix.util
Module MatrixAttributes
    Sub Main()

        Dim context As Context
        Dim command As String()

        command = System.Environment.GetCommandLineArgs()
        If command.Length() < 7 Then
            Console.WriteLine("Usage:" + vbCrLf + " VBMATRIXATTRIBS.EXE host:port user"
password type name revision vault")
            Console.WriteLine("Example:" + vbCrLf + " VBMATRIXATTRIBS.EXE localhost:7001
creator """" ECR 4000 """ mxSample")
            Return
        End If
        Try
            context = New Context("", command(1))
            context.setUser(command(2))
            context.setPassword(command(3))
            context.connect()
        Catch ex As MatrixException
            Console.WriteLine("Connection failed: " + ex.ToString())
            Return
        End Try
        Try
            Dim vault As String
            Dim type As New String(command(4))
            Dim name As New String(command(5))
            Dim rev As New String(command(6))
            If (command.Length() = 8) Then
                vault = command(7)
            Else
                vault = ""
            End If
            Dim bo As New BusinessObject(type, name, rev, vault)
            bo.open(context)
            Dim attributes As BusinessObjectAttributes
            Dim attriblist As AttributeList
            Dim attribitr As AttributeItr
            attributes = bo.getAttributes(context)
            attriblist = attributes.getAttributes()
            attribitr = New AttributeItr(attriblist)
            Dim attrib As matrix.db.Attribute
            Console.WriteLine("Business object attributes: " + type + " " + name + " "
+ rev + " " + vault + vbCrLf)
            While attribitr.next()
                attrib = attribitr.obj()
                Console.WriteLine(attrib.getName + ": " + attrib.getValue)
            End While
            bo.close(context)
        Catch ex As MatrixException
            Console.WriteLine("Exception: " + ex.ToString())
        End Try
        context.shutdown()
    End Sub
End Module
```

Building a Sample Application

You can build a console application using the VB.NET module shown above. The application will take the object type, name, and revision you specify and display its attributes on the console.

1. Copy the VB.NET sample code in the [Sample Code](#) section and save it as `MatrixAttributes.vb`.
2. From the command line, type:

```
vbc.exe /out:VBMATRIXAttribs.exe /  
reference:eMatrixClientXML.dll,  
  
FULLPATH\vjslib.dll MatrixAttributes.vb
```

Notes:

- `VBMATRIXAttribs.exe` is the output file to be created.
- You must specify the full path for `vjslib.dll`.
- `MatrixAttributes.vb` contains the sample VB.NET code.

The following files should be in the current directory:
`VBMATRIXAttribs.exe`

`eMatrixClientXML.dll`

`MatrixAttributes.vb`.

3. Run `VBMATRIXAttribs.exe` with the following command line arguments:

Host:port user password type name revision vault (optional)

For example:

```
VBMATRIXAttribs.exe localhost:7001 creator sparks2 ECR 4000 1  
mxSample
```

The above command will display attributes for object type `ECR`, name `4000`, revision `1` using the `mxSample` vault. The host, port, username, and password are used to access the object data.

.NET Limitations

This section describes limitations when using .NET with the ENOVIA Live Collaboration Applet Studio Customization Toolkit.

The following topics are discussed:

- [Object Serialization](#)
- [Studio Customization Toolkit in Multi-threaded Applications](#)

Object Serialization

Objects serialized under JDK 1.1.4 can be deserialized in Visual J# only if the fields of the objects are primitive types or `java.lang.String`. An object referencing any other classes in JDK level 1.1.4 packages cannot be deserialized. Of course, any object serialized using Visual J# can be deserialized in Visual J#.

Due to the `java.io.Serializable` compatibility problem, the FCS feature for .NET is disabled.

Additionally, `JPO.invoke` will work from .NET only if objects passed to and returned from a JPO can be safely serialized and deserialized between environments.



Studio Customization Toolkit in Multi-threaded Applications

Applications may be multi-threaded. However, it is recommended that all Studio Customization Toolkit calls be made from one thread.

File Collaboration Server

The File Collaboration Server (FCS) provides location, checkout, and copy operations on the data managed by the ENOVIA Live Collaboration Server. This section provides details about how the FCS works.

In this section:

- [File Collaboration Server](#)
- [FCS Architecture](#)
- [Stores, Sites, and Locations](#)
- [FCS Synchronization](#)
- [File Checkins](#)
- [About File Checkouts](#)
- [Replication](#)
- [Business Object Proxies \(BOPs\)](#)
- [FcsClient](#)
- [Large Files](#)
- [File Stream Filters](#)
- [Debugging](#)
- [Custom JSP Pages for FCS](#)
- [FCS Network Security](#)

File Collaboration Server

The File Collaboration Server (FCS) provides location, checkout, and copy operations on the data managed by the ENOVIA Live Collaboration Server.

Related Topics:

[About FCS Network Security](#)
[About FCS Network Security Views](#)
[Implementing Out-of-the-Box Authentication Management](#)
[Implementing Custom Authentication Management](#)
[FCS Network Security Error Messages](#)

The FCS brings the ENOVIA Live Collaboration Server file storage functions closer to users on the same LAN segment or across the WAN. The FCS provides these advantages:

- It does not require a database connection.
- It serves multiple ENOVIA Live Collaboration Servers.
- It does not require multiple copies of applications.
- It does not require any configuration. You enable the FCS using system or MQL commands.
- It provides support for both Single Sign On (SSO) and authentication.

The prerequisites and requirements for the FCS are the same as those for the ENOVIA Live Collaboration Server. See "Prerequisites and Requirements" in the *ENOVIA Live Collaboration Installation Guide* for details.

Note: The FCS was briefly referred to as eFCS. This name is no longer used.

FCS Architecture

This section describes the FCS architecture.

In this section:

- ❑ [FCS Architecture/Configuration](#)
- ❑ [Other Supported Configurations](#)
- ❑ [Tickets and Receipts](#)
- ❑ [Added Security](#)

FCS Architecture/Configuration

The FCS is installed with the ENOVIA Live Collaboration Server and enabled by default. This means there is no additional installation needed. To use an FCS, you define an FCS URL for each store and location you want to enable. The FCS URL points to the store or location server containing the actual file.

You must define the FCS URL for FCS file replication. The FCS URL is not required for file checkin and checkout operations.

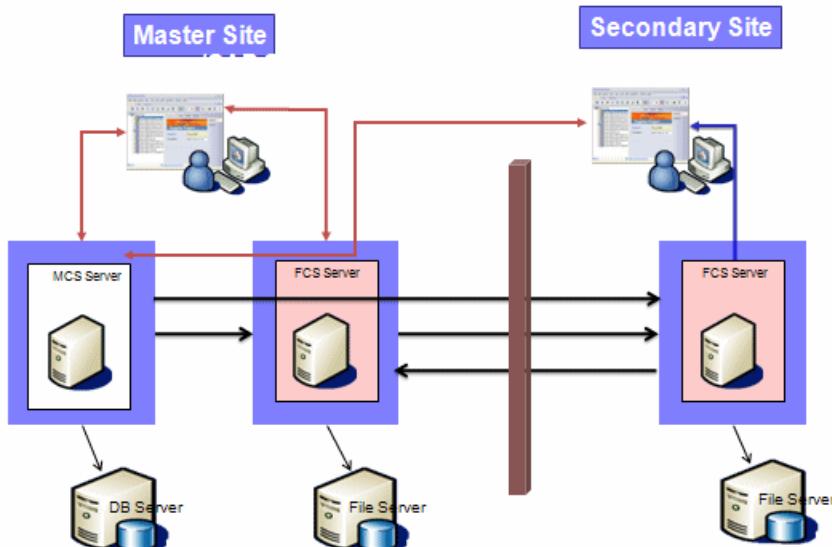
When an FCS URL is not defined, the system uses the URL of the local MCS. Since there are two ways to connect to the ENOVIA Live Collaboration Server, it is important to note that the MCS URL can only be inferred when connecting through the application server, usually at port 8080. The other way to connect is through the server, usually at port 1099, but an MCS URL cannot be inferred when connecting this way because an MCS URL is not defined. This means if you are connecting through the server and need to perform FCS operations that require an FCS URL, you must manually define the FCS URL.

You specify settings in web.xml or the framework.properties file to control various parts of the FCS system. You should add all settings to web.xml instead of framework.properties. If a property is set in both framework.properties and web.xml, the value in web.xml is used. The various parts of the system controlled include:

- The length of time a ticket or receipt is valid.
- How many tickets or receipts the system retains before deleting them.

FTP is not required and should not be used for WAN transfer. FTP should only be used for local machine transfer. All communication between the MCS and FCS must be via HTTP.

This diagram illustrates a typical configuration with a single ENOVIA Live Collaboration Server or Main Collaboration Server (MCS) and one or more FCSs.



FCS Component	Description
Client/Browser	The client browser requests a file storage operation. If the FCS is configured on this LAN, then the FCS intercepts the request and processes it based on the ticket passed by the client. The FCS supports: Browser-based and HTTP clients.
File Protocol	The system uses HTTP/S for the checkin request and all encrypted ticket/receipt transfers. The FCS checks in and checks out files from the store or location. The FCS uses HTTP/S to get the store/location information from the ENOVIA Live Collaboration Server. The ENOVIA Live Collaboration Server communicates with the database over a LAN connection. The connection to the database, via a WAN connection, is eliminated.
MCS	Systems are typically configured for a single MCS and one or more FCSs servicing the local requests.
Tickets and Receipts	Tickets and receipts are the vehicle for moving information between the ENOVIA Live Collaboration

	Server and the FCS. They provide added security.
Triggers	<p>The FCS uses triggers with Java Programming Objects (JPOs) during checkin and checkout operations. Supported triggers include:</p> <ul style="list-style-type: none"> Check--Fires before an event Action--Fires after an event When a trigger fails, the system rolls back the transaction. The Override trigger type is not supported by the FCS.

ENOVIA Live Collaboration uses HTTP/S to transfer files between FCS locations and stores. This includes all communication between the MCS and FCS. If an FCS location or store is configured with the FTP protocol, FTP will only be used locally for the FCS server to talk to its own FTP server. FTP should only be used for local machine transfer.

If you receive an error message that begins with "MLC Warning," it means you are running different versions of the MCS and FCS at the same time.

The amount of programming necessary with the FCS depends on the type of environment:

- If you want to use the AppletXML and Studio Customization Toolkit, setup is minimal. Java classes for business object checkin and checkout are enabled for FCS. You simply define which stores/locations have FCS URLs.
- When using the Studio Customization Toolkit *without* AppletXML (running in the ENOVIA Live Collaboration Server), an FCS is normally not used. You can make the Studio Customization Toolkit use an FCS for checkin/checkout without AppletXML by setting the environment variable `MX_FCS_ENABLED` to `TRUE`. When set to `FALSE`, FCS is not used.
- If you want to use ENOVIA products, setup is minimal. Application JSP pages are enabled for an FCS. You simply define which stores/locations have FCS URLs. All others will default to the URL of the ENOVIA Live Collaboration Server.
- For installations which have their own JSP-based applications, see [Your Own Checkout](#) and [Using the com.matrixone.fcs.mcs](#).

Other Supported Configurations

You can have additional configurations with multiple FCS servers enabled in different locations servicing the same clients.

This configuration requires replication, see [Replication](#) for more information

If you would like to use an FCS in a Single Sign-on (SSO) environment, you must add each MCS and FCS in your environment to the access list that excludes them from authentication checks. Please be aware that you cannot use a remote thick client with SSO. A local thick client can be used with SSO since it will be on an MCS that should be on the access list.

Tickets and Receipts

The FCS uses tickets and receipts as container vehicles to move and describe metadata.

For debugging purposes, FCS tickets do not contain serialized objects. FTP passwords in tickets remain encrypted. Tickets define:

- The file operation. The operation for the FCS to perform for the client
- Common data. Defined by the schema/DTD for the ticket.
- One or more requests for files. The files the FCS performs the file operation. For example, one ticket might contain a checkin request, common data for the checkin request, and five individual requests for files.

Valid file operations supported by the tickets are:

- Checkout
- Checkin
- Copy--The ticket created for a copy operation contains an embedded checkout ticket which is used by the destination FCS to perform a checkout operation on the source FCS.
- Delete--The delete operation is handled by a background thread. After a transaction completes, the delete operation remains in a job cue and completes later.

When the FCS receives a ticket, it processes the data and generates a Receipt. It only generates a receipt when the file operation is a checkin or a copy. A receipt contains the information about file format, whether to append the data or not, what kind of lock to use during the operation, the override, the file name, and the file size. The FCS then sends the receipt to the ENOVIA Live Collaboration Server to begin the file operation.

Once the ENOVIA Live Collaboration Server receives the receipt, it processes and commits the information. The transaction is considered completed once the ENOVIA Live Collaboration Server update completes. If the receipt processing on the ENOVIA Live Collaboration Server encounters an error, the system rolls back the ENOVIA Live Collaboration Server file operation. A rollback occurs when the ENOVIA Live Collaboration Server cleans up the files managed by the FCS. The FCS is stateless so it has no notion of success or failure on the ENOVIA Live Collaboration Server side.

If the checkin or copy operation fails while the FSC is writing it to disk, the FCS:

- Performs a cleanup operation.
- Does not generate a receipt.
- Generates an error condition
- Depending on the client type, the FCS performs a different action.
- Browser-based clients--the FCS redirects to an error page.
- HTTP clients (Java, .NET, and so on)--The FCS returns an HTTP error code.
- Determines the client type based on the caller that creates the initial request via the Ticket. The API determines the client type.
- Browser-based clients use these APIs: com.matrixone.fcs.http.HttpCheckin/ HttpCheckout class
- HTTP clients use these APIs: com.matrixone.fcs.mcs.Checkin*/Checkout.

Added Security

Tickets and receipts provide additional security.

Tickets and receipts are secure because they are:

- Single use
- Encrypted
- Time sensitive--expire after five minutes. When multiple machines are involved in the network, it's possible that the FCS and ENOVIA Live Collaboration Server clock settings differ. To avoid this problem, the FCS does a callback operation to synchronize time differences so the ticket expiration is accurate.

Stores, Sites, and Locations

Stores, Sites, and Locations are an integral part of how the ENOVIA Live Collaboration Server and the FSC work. The following section describes how the FCS uses them.

The following topics are discussed:

- [Stores](#)
- [Sites](#)
- [Locations](#)
- [Synchronization](#)

Related Topics:

[Synchronizing Captured Stores](#)
[FCS Compressed Synchronization](#)

Stores

Stores are storage locations for checked-in files. FCS exclusively manages stores. To create a new store to replace an existing one, you must change the policies referencing the old store to reference the new one. This ensures that all new files checked into objects governed by the policies are placed in the new store.

With replicated stores, a store can be distributed using asynchronous replication.—where data is duplicated to all mirrored locations only when commands are passed telling the database to do so. In this case, when files are checked in they are written to the store location that is part of the user's preferred site, or if no match is found, to the store's default location. Files are not replicated to alternate locations until specifically told to do so.

A multi-site installation is an ENOVIA V6 deployment containing at least two sites and two locations with replicated stores. See also [Locations](#). Store replication makes sense when an ENOVIA V6 deployment spans through network links with low bandwidth or high latency, or both.



Sites

Sites comprise a set of locations. A site is an administrative object that can be associated with a person or group object. When associated with a person, the site defines the list of locations preferred by a particular person. When associated with a group, the site defines the list of locations preferred by all members of the group. Only System Administrators can create sites. Refer to the *Business Modeler Guide* or *MQL Guide* for more information.

When files are accessed via an ENOVIA Live Collaboration Server, the server's initialization file (enovia.ini) should contain the following to set the preference for all of its users:

`MX_SITE_PREFERENCE = SITENAME`

where SITENAME is the name of the site object.

The `MX_SITE_PREFERENCE` is optional, it overrides the site preferences. However, this is not normally set. This setting overrides the setting in the person, group, or role definition for the site preference, and should be set to the site local to the server. Additionally, you can configure a File Collaboration Server to optimize the performance of file checkin and checkout in a replicated environment.



Locations

Think of a location as another store--it is defined as an FTP/NFS or UNC location. The same rules apply as in specifying a store.

Locations contain alternate host, path and FTP information for a captured store. The host, path and FTP information in the store definition is considered to be the *default* location for the store, while any associated location objects identify *alternate* file servers.

Each location must have a name unique in the database. The name can be up to 127 characters and can contain spaces. Because names are so flexible, you should assign a name that has meaning to both you and your users.

After you have created locations, assign them to captured stores. Any number of locations can be associated with a captured store. Locations can be shared by multiple stores. However, it is recommended that if one store is hashed, all stores that use any of its locations are also hashed especially if full text search is used.

You can define a protocol for a location as: ftp, ftps, and nfs. If a location does not have a protocol specified, ENOVIA Live Collaboration looks at other object attributes to determine which protocol was intended. If the host is localhost (or empty), the system uses the file (local file system) protocol. If the host is NOT localhost and a username/password is given, then the location uses ftp. If the host is NOT localhost and there is NO username/password, the location uses file.



Synchronization

Synchronization of a store using one of its locations is the process of updating the store's files by using the location's files. Synchronization of a location using its store is the process of updating the location's files by using the store's files. Synchronization on demand is a synchronization process that is triggered by an ENOVIA V6 checkout user request. If the user

checks out at least one file that is not up to date in their preferred location, this checkout request triggers a synchronization of that preferred location. [FCS Compressed Synchronization](#) can improve FCS server synchronization performance in a WAN environment. For MQL commands to enable and configure compressed synchronization, see the *MQL Guide*.

FCS Synchronization

Synchronization ensures that users can access the latest versions of files when checking out or viewing files in a replicated store environment. Compressed synchronization is supported for WAN environments in which network latency exceeds 1 ms.

In this section:

-  [Synchronizing Captured Stores](#)
-  [FCS Compressed Synchronization](#)

Synchronizing Captured Stores

Synchronization ensures that users access the latest versions of files when checking out or viewing files in a replicated store environment.

The following topics are discussed:

- [Synchronization Methods](#)
- [Automatic Synchronization](#)
- [Manual Synchronization](#)
- [Proxy Server for FCS Synchronization](#)

Related Topics:

[ECS Compressed Synchronization](#)

Synchronization Methods

Synchronization occurs in one of two ways:

- ENOVIA Live Collaboration automatically synchronizes files for a user's preferred location during checkout and open for viewing operations.
- System Administrators can use MQL commands to synchronize files for a business object or for a store to ensure that all locations contain the most recent copy of the files.

For complete details on synchronization, see the "Building the System Basics" chapter in the *MQL Guide*.



Automatic Synchronization

Synchronization occurs automatically when a user checks out or opens a file for viewing and the file at the user's preferred location is not the latest version. In such a case, ENOVIA Live Collaboration copies the latest version of the file to the user's preferred location and then checks out or opens the local file.



Manual Synchronization

In order to publish newly checked-in files to other locations associated with the store, System Administrators must use the MQL sync command against either the business object or the store. For details, see the "Building the System Basics" chapter in the *MQL Guide*.



Proxy Server for FCS Synchronization

A proxy server may be setup for FCS synchronization operations where file transfers occur between locations inside and outside of ENOVIA Live Collaboration. Proxy servers allow firewalls to be passed more easily and are especially important for companies that exchange data between divisions and subsidiaries or with partners and clients.

All FCS synchronization operations use HTTP protocol (FTP is no longer used) to handle file transfers. A proxy server setup requires that all the nodes are able to communicate with each other over HTTP.

To setup a proxy server, configure the ENOVIA File Collaboration Server as a proxy by adding the "-Dhttp.proxyHost=proxyHost" and "-Dhttp.proxyPort=proxyPort" options to mxEnv.sh or the application server startup script JAVA_OPTIONS environment variable. For the MQL client, add them to MX_JAVA_OPTIONS in the enovia.ini file for the Studio Modeling Platform or startup script for UNIX.

For example:

```
MX_JAVA_OPTIONS="-Xss512k -Xms640m -Xmx640m -Dmatrix.debug=true  
-Dhttp.proxyHost=Host -Dhttp.proxyPort=Port"  
JAVA_OPTIONS="${MX_JAVA_OPTIONS}"
```

For HTTPS, use https.proxyHost and https.proxyPort.

ENOVIA Live Collaboration Server currently does not support HTTP authentication on client side proxies.

FCS Compressed Synchronization

The `zipsync`, or compressed synchronization, function improves performance of all FCS synchronization operations in a WAN environment.

Related Topics:
[Stores, Sites, and Locations](#)
[File Collaboration Server](#)

The `zipsync` function enables compression of FCS data streams exchanged during synchronization or replication operations. Compressed synchronization reduces the synchronization-process-elapse time as soon as wide area network latency exceeds 1 ms. Compression can be controlled through a set of simple MQL commands.

In V6R2011, compression of synchronization streams had initially been introduced with an extension of the "sync businessobject mql" command and rules for compressed synchronization, which were added to the FCS Synchronization Server. With V6R2011x, the `zipsync` function further extends and automates the compression of synchronization streams for all synchronization scenarios, including:

- Explicit synchronization performed using the MQL commands `sync store`, `sync businessobjectlist`, and `sync businessobjectlist`; and the `FCSSyncServer` tool.
- Implicit synchronization or synchronization-on-demand.

A synchronization process occurring between two locations, or a store and a location, is considered to be WAN-wide (in other words, occurring in a WAN-environment) if the two locations, or the store and the location, belong to distinct sites. For example, assume there is a store with two locations, A and B. Location A belongs to site A and location B belongs to site B. Any synchronization that occurs between location A and location B is considered to be WAN-wide. A synchronization that occurs between the store and location A, or the store and location B, is also considered to be WAN-wide because the store belongs to the central site by default.

If compressed synchronization is enabled, any WAN-wide FCS synchronization is compressed. More precisely, the files exchanged during the synchronization process are compressed. The FCS server that sends the file compresses the data (if it is not already compressed), and the FCS server that receives the file decompresses it. The compression function does not compress files that are already compressed, as this would cause a performance loss. FCS determines whether a file is already compressed or not by examining the file extension.

With V6R2011x, compressed synchronization is enabled by default. In other words, after any V6R2011x setup or upgrade, compressed synchronization will be turned on.

MQL provides commands to activate and configure compressed synchronization. For example, you can configure the list of file extensions that specify whether files are compressed by using MQL commands. For details, see the following sections in the *MQL Guide*:

- Chapter 4, "Building the System Basics" > section "Synchronizing Captured Stores" > subsection "FCS Compressed Synchronization"
- Chapter "Reference Commands" > section "set system Command"

File Checkins

This section describes the methods the FCS uses for checkin operations.

In this section:

- [About File Checkins](#)
- [One-Part Checkin](#)
- [Two-Part Checkins](#)
- [Checkin with Correlate](#)

About File Checkins

The FCS supports two types of checkin: one-part checkin is employed when the business object the file is checked into does not yet exist, and two-part checkin is used when the business object already exists. FCS allows override triggers, but only when FCS is disabled during the override trigger operation.

Checking in a set of files to the FCS involves four steps:

1. Creating a ticket corresponding to the number of files.
2. Using the ticket, uploading the files into the FCS, and receiving a receipt.
3. Creating the object proxies that materialize the association between a Business Object and a file previously uploaded.
4. Validating the check in operation using the receipt and the list of proxies.

Any Java client application that will perform FCS checkin or checkout operations, must now explicitly reference the FcsClient.jar instead of the eMatrixServletRMI.jar. The following error will occur if the Java client is not referencing the FcsClient.jar: <java.lang.NoClassDefFoundError: com/matrixone/client/fcs/InputStreamSource_2>

Before a checkin/checkout operation to/from a store or location, verify that the FCS server defined in the FCS_URL on that store or location is running. If not, the operation will fail. Additionally, you should verify factors that may also affect the operation, like a LAN line being down.

If several users try to check in files simultaneously, some of the checkins may fail with an FCS error. This can be avoided by increasing the thread pool size of the application server. The following table provides information on adjusting thread pool size in various application servers.

Application Server	Adjusting Thread Pool Size
WebSphere	See http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/uejb_rthrd.html .
WebLogic	Tune the pool size by providing pool sizes (such as pools for JDBC connections, Stateless Session EJBs, and MDBs) that maximize concurrency for the expected thread utilization. <ul style="list-style-type: none">• For WebLogic Server v9.0 and higher—A server instance uses a self-tuned thread pool. The best way to determine the appropriate pool size is to monitor the pool's current size, shrink counts, grow counts, and wait counts. See "Thread Management." Tuning MDBs are a special case. See "Tuning Message-Driven Beans."• For earlier releases—in general, the number of connections should equal the number of threads that are expected to be required to process the requests handled by the pool. The most effective way to ensure that the pool size is right is to monitor it and make sure that it does not shrink and grow. See "How to Enable the WebLogic 8.1 Thread Pool Model."
Sun Java System Application Server	See http://docs.sun.com/app/docs/doc/819-3681/abefm?l=En&a=view .
Tomcat	See http://tomcat.apache.org/tomcat-6.0-doc/config/executor.html .

One-Part Checkin

In one-part checkin, the business object that the file will be checked into typically does not yet exist. This type of checkin is supported by ENOVIA products.

The following topics are discussed:

- [Sequence](#)
- [Your Own One-Part Checkin](#)
- [One-Part Checkins Using the com.matrixone.fcs.mcs](#)

Sequence

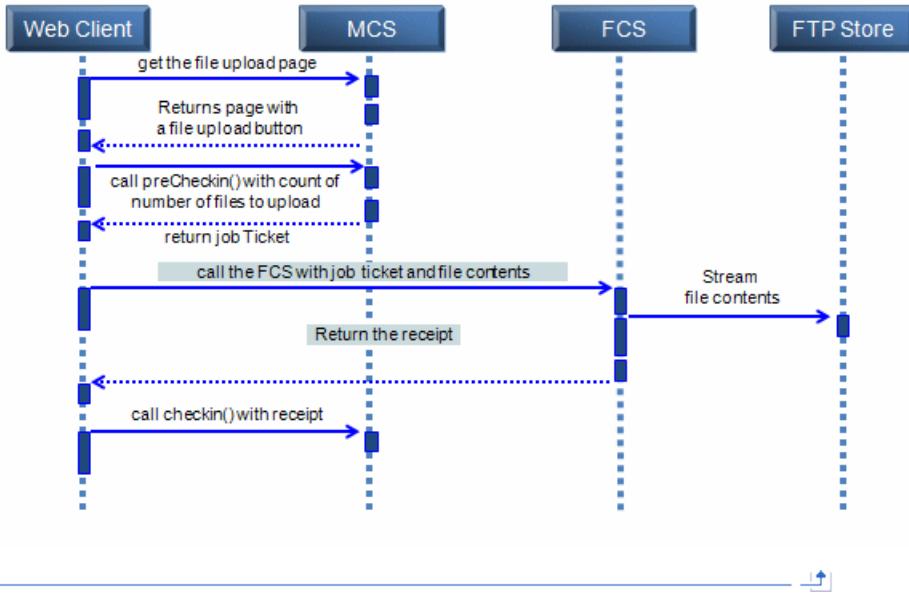
The application uses the preCheckin() method to obtain a ticket with no OID and format. The FCS then processes the file(s). Upon ticket receipt, the application calls the ENOVIA Live Collaboration Server checkin methods and any triggers are fired. Here is the sequence for one-part checkin:

1. Get the ticket using preCheckin()
2. Transfer the file.
3. Call checkin()
 - a. Check trigger is fired.
 - b. Meta data is written.
 - c. Action trigger is fired.

If any triggers in the above sequence are blocked, the file that has already been transferred to the store/location will be deleted by the MQL.

One Part Checkin Sequence

One Part Checkin Interaction Diagram Using Web Browser (with FTP)



Your Own One-Part Checkin

To write a client that performs a one-part checkin, use the Precheckin.dolt() and Checkin.dolt() methods. The classes that you use depend on the kind of client you are writing. The dolt() methods are available in these packages:

- com.matrixone.fcs.mcs--Use this when the client performing the checkin is located on the MQL/FCS server.
- com.matrixone.fcs.http.Http--Use this when the client performing the checkin is a browser client.

One-Part Checkins Using the com.matrixone.fcs.mcs

First call the com.matrixone.fcs.mcs.Precheckin.doIt() method which has two variants. Use the second variant when you want to perform a location override.

```
Static TicketWrapper doIt(Context?ctx, String?store,
String?connectString, int?cnt)
static TicketWrapper doIt(Context ctx, String?store,
String?locationOverride, String?connectString, int?cnt)
```

The parameters for both variants are described in the following table:

Argument	Description
Context	ENOVIA Live Collaboration context
Store	A valid location
LocationOverride	The URL to the location where the checkout should be performed

Once you have the ticket, use the com.matrixone.fcs.mcs.checkin.doIt() to perform the actual checkin operation. It has this syntax:

```
public static void doIt(Context?ctx, String?receiptValue,
String?store, ArrayList?list) throws MatrixException
```

The arguments are described in the following table:

Argument	Description
Context	The ENOVIA Live Collaboration context
receiptValue	
Store	A valid location
List	The list of proxies given during the last step is in the exact same order as the related files have been uploaded. If they are not, the wrong files may be associated to the VPLM metadata, leading to possible data issues that will be detected later on when trying to read the data and will be very difficult to fix.

The system will check for the following errors when listing proxies. These situations will lead to exceptions that will rollback the transaction.

- The count of proxy list entries with file names does not match the size of the proxy list.
- There are duplicate file names in the proxy list
- There are duplicate file names in the receipt.
- The file is mentioned in the receipt, but no corresponding proxy found.
- The proxy is passed in, but no corresponding entry in the receipt.

First use the com.matrixone.fcs.http.HTTPPrecheckin.doIt() method. It looks like this:

```
public static TicketWrapper doIt(Context?ctx, String?store,
int?fileCount, String?processingPage, String?targetPage,
String?errorCode, javax.servlet.http.HttpServletRequest?req,
javax.servlet.http.HttpServletResponse?res) throws
MatrixException
```

The arguments are described in the following table:

Argument	Description
Ctx	The ENOVIA Live Collaboration context
Store	The store name
fileCount	The number of files to checkin
ProcessingPage	The page for the checkin end
targetPage	The final destination page
errorCode	The error page
Req	The servlet request. It must contain params of bold, fileName, format, append, and unlock. The system throws an exception if the count does not match.
Res	The servlet response

This example shows how to use the `Http.precheckin.doIt()` method to obtain a ticket.

```
.  
. .  
  
String processingPage = "/fcs/checkinMCS.jsp";  
String targetPage = "/fcs/thankYou.jsp";  
String errorPage = "/fcs/errorPage.jsp";  
Context ctx = Framework.getFrameContext(session);  
int numFiles = new  
Integer(request.getParameter("numFiles")).intValue();  
TicketWrapper ticket = HttpPreCheckin.doIt(ctx, store,  
numFiles, processingPage,  
targetPage, errorPage, locationOverride, request, response);  
String ticketStr = ticket.getExportString();  
String actionURL = ticket.getActionURL();  
. .  
. .  
  
<html>  
<body>  
<script>  
var fcsForm = parent.frames["fcsForm"].document.forms[0];  
fcsForm.<%=McsBase.resolveFcsParam("jobTicket")%>.value='<=%=tic  
ketStr%>';  
fcsForm.action='<=%=actionURL%>';  
fcsForm.submit();  
</script>  
</body>  
</html>  
. .  
. .
```

Once you have the ticket, construct the business object and then perform the actual checkin using this `doIt()` method:

```
public static void doIt(Context?ctx, String?store,  
java.util.ArrayList?list,  
javax.servlet.http.HttpServletRequest?req,  
javax.servlet.http.HttpServletResponse?resp) throws  
MatrixException
```

The arguments are described in the following table:

Argument	Description
Ctx	The ENOVIA context
Store	The store name
List	The list of proxy files
ProcessingPage	The page for the checkin end
targetPage	The final destination page
errorPage	The error page
Req	The servlet request. It must contain params of boid, fileName, format, append, and unlock. The system throws an exception if the count does not match.
Resp	The servlet response

The following code illustrates how to create a new business object proxy passing in the object id, file format. It appends the checkin file and leaves object in the unlocked state after the checkin completes.

```
Context ctx = Framework.getFrameContext(session);  
String unlock = request.getParameter("unlock");  
String append = request.getParameter("append");  
String user = request.getParameter("user");
```

```
System.out.println("User supplied parameter: " + user);
ArrayList list = new ArrayList();
for (int t = 0;t < 5;t++)
{
BusinessObjectProxy bop = new
BusinessObjectProxy(obj.getObjectId(),format,
append.equals("true"),
unlock.equals("true"));
list.add(bop);
}
String receiptValue =
request.getParameter(McsBase.resolveFcsParam("jobReceipt"));
HttpCheckin.doIt(context, store,list,request,response);
```

Two-Part Checkins

Two-part checkin is used when the business object already exists. When the ticket (containing the OID and the format) is obtained, the check trigger is fired. FCS processes the file and returns the receipt. The application calls the checkinEnd() method.

The following topics are discussed:

- [Sequence](#)
- [Your Own Two-part Checkin](#)
- [Two-Part Checkins Using com.matrixone.fcs.mcs](#)
- [Two-part Checkins Using com.matrixone.http.Http](#)

Sequence

Below is the sequence for two-part checkin:

Get the ticket using checkinStart(). Check trigger is fired.

Transfer the file.

Call checkinEnd()

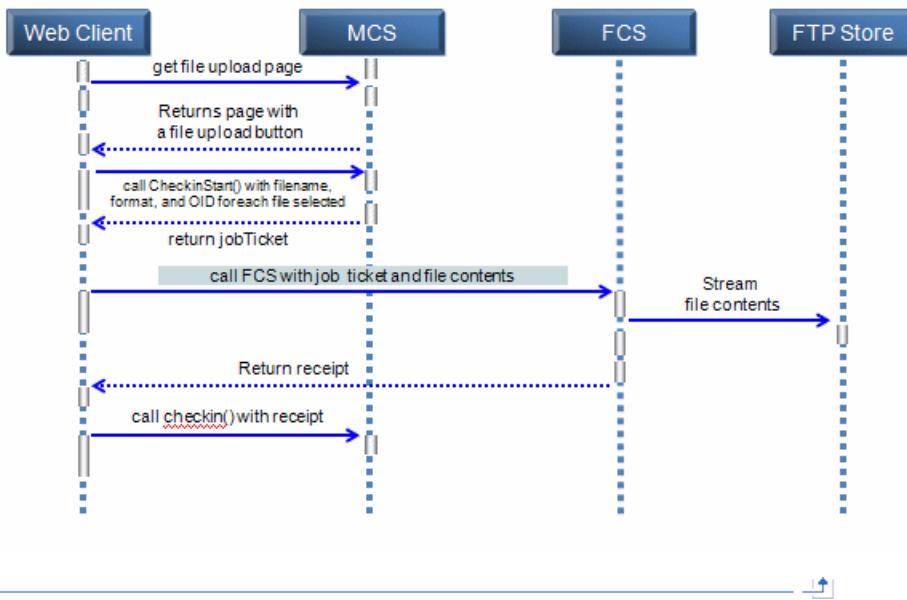
Meta data is written.

Action trigger is fired.

The following diagram illustrates two-part checkin:

Two Part Checkin Sequence

Two Part Checkin Interaction Diagram Using Web Browser (with FTP)



Your Own Two-part Checkin

To write a client that performs a two-part checkin, use the doIt() method of the CheckinStart and CheckinEnd classes. The classes that you use depend on the kind of client you are writing. The doIt() methods are available in these packages:

- com.matrixone.fcs.mcs--Use this when the client performing the checkin is located on the Live Collaboration Server/FCS server.
- com.matrixone.fcs.http.Http--Use this when the client performing the checkin is a browser client.

Two-Part Checkins Using com.matrixone.fcs.mcs

The Checkin.start.doIt() method has four variants. Use this variant when the files are checked into the default store:

```

public static TicketWrapper doIt(Context?ctx,
String?connectString,
java.util.ArrayList?list) throws MatrixException
Use this variant to specify the checkin store:
public static TicketWrapper doIt(Context?ctx,
String?connectString,
String?store, java.util.ArrayList?list) throws MatrixException
Use this variant to allow the user to override the store location:
public static TicketWrapper doIt(Context?ctx,
String?connectString,
String?store, String?locationOverride,
java.util.ArrayList?list) throws
MatrixException

```

The arguments are:

Argument	Description
Ctx	The ENOVIA context
connectString	
Store	The checkin store
locationOverride	The checkin store's location
List	The list of proxies given during the last step is in the exact same order as the related files have been uploaded. If they are not, the wrong files may be associated to the VPLM metadata, leading to possible data issues that will be detected later on when trying to read the data and will be very difficult to fix.

The system will check for the following errors when listing proxies. These situations will lead to exceptions that will rollback the transaction.

- The count of proxy list entries with file names does not match the size of the proxy list.
- There are duplicate file names in the proxy list
- There are duplicate file names in the receipt.
- The file is mentioned in the receipt, but no corresponding proxy found.
- The proxy is passed in, but no corresponding entry in the receipt.

To complete the checkin, call the CheckinEnd.doIt() method using one of these variants:

```

public static void doIt(Context?ctx, String?receiptValue)
throws java.lang.Exception
public static void doIt(Context?ctx, String?store,
String?receiptValue)
throws MatrixException

```



Two-part Checkins Using com.matrixone.http.Http

The HttpCheckinStart.doIt() method has the two variants shown below. Use the second variant when you want to allow the user to override the checkin location.

```

public static TicketWrapper doIt(Context?ctx,
String?processingPage,
String?targetPage, String?errorCode, String?store,
String[]?boids,
String[]?fileNames, String[]?formats, String[]?append, String[]?unlock,
javax.servlet.http.HttpServletRequest?req,
javax.servlet.http.HttpServletResponse?res) throws
MatrixException
public static TicketWrapper doIt(Context?ctx,
String?processingPage,
String?targetPage, String?errorCode, String?store,
String?locationOverride, String[]?boids, String[]?fileNames,
String[]?formats, String[]?append, String[]?unlock,
javax.servlet.http.HttpServletRequest?req,
javax.servlet.http.HttpServletResponse?res) throws
MatrixException

```

The arguments are described in the following table:

Argument	Description
Ctx	The ENOVIA context
processingPage	The page to use for the checkin end. This is the page where the checkin.end() is called from. This page should perform any user processing. Users' parameters will be available on this page.
targetPage	The final destination page.
errorPage	The error page.
store	The name of the store.
locationOverride	The name of the store for an override operation.
Req	The HTTP servlet request. It must have these parameters or an exception is thrown: boid, filename, format, append, and unlock.
res	The HTTP servlet response.

To complete the checkin, call the `HttpCheckinEnd.doIt()` method which has this syntax:

```
public static void doIt(Context?ctx, String?store,
javax.servlet.http.HttpServletRequest?req,
javax.servlet.http.HttpServletResponse?resp)
throws MatrixException
```

The following example illustrates how to use the `HttpCheckinStart.doIt()` method:

```
String targetPage = "/fcs/thankYou.jsp";
String errorPage = "/fcs/errorPage.jsp";
Context ctx = Framework.getFrameContext(session);
int numFiles = new
Integer(request.getParameter("numFiles")).intValue();
String[] fileNames = request.getParameterValues("fileName");
String[] formats = new String[numFiles];
String[] append = new String[numFiles];
String[] unlock = new String[numFiles];
String[] oids = new String[numFiles];
String[] files = new String[numFiles];
for (int t=0;t<numFiles;t++)
{
    formats[t] = format;
    oids[t] = oid;
    append[t] = appendDefault + "";
    unlock[t] = unlockDefault + "";
    files[t] = fileNames[t];
}
TicketWrapper ticket = HttpCheckinStart.doIt(ctx,
processingPage,
targetPage, errorPage, store, locationOverride,
oids, files, formats, append, unlock, request, response);
String ticketStr = ticket.getExportString();
String actionURL = ticket.getActionURL();
%>
<html>
<body>
<script>
var fcsForm = parent.frames["fcsForm"].document.forms[0];
fcsForm.<%=McsBase.resolveFcsParam("jobTicket")%>.value='<=%tic
ketStr%>';
fcsForm.action='<%=actionURL%>';
fcsForm.submit();
</script>
</body>
</html>
```

This example shows how to complete the checkin with the `HttpCheckinEnd.doIt()` method.

.

```
<%  
Context ctx = Framework.getFrameContext(session);  
HttpCheckinEnd.doIt(ctx, store, request, response);  
%>
```

Checkin with Correlate

The Correlate feature on the checkin method supports "format" as a formal parameter.

Formats are specified by defining form variables. Form variables are defined before the file content section and use the following format:

`__fcs_format_x`

Where `x` is the file count. Variable names are constructed as follows:

```
com.matrixone.client.fcs.FcsClient.getFormatFieldName(count);
```

In most cases, the format parameter is optional. The format parameter must be set in cases where the BOP List contains reference to the same business object with the same filename. For example:

```
(from com.matrixone.fcs.mcs.Checkin)
    public static void doIt(Context ctx, String receiptValue,
String store,
                        ArrayList list,boolean correlate) throws
MatrixException {
    try {
        process(ctx, receiptValue, store, list,correlate);
    } catch (Exception ex) {
        throw new MatrixException(ex.getMessage());
    }
}
```

About File Checkouts

This section describes how to write FCS clients to perform checkout operations.

The following topics are discussed:

- [Sequence](#)
- [Checkout Locations](#)
- [FCS Zip Support](#)
- [Bulk Document Checkout](#)
- [Your Own Checkout](#)
- [Using the com.matrixone.fcs.mcs](#)
- [Using the com.matrixone.fcs.http Methods](#)
- [Sample Checkout Code](#)
- [Using the com.matrixone.fcs.fcs Methods](#)

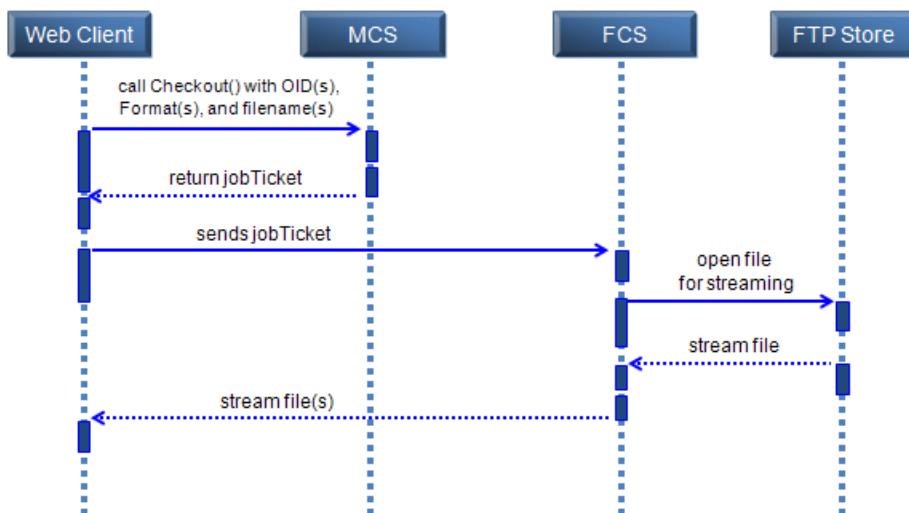
Sequence

Any Java client application that will perform FCS checkin or checkout operations, must now explicitly reference the FcsClient.jar instead of the eMatrixServletRMI.jar. The following error will occur if the Java client is not referencing the FcsClient.jar: <java.lang.NoClassDefFoundError: com/matrixone/client/fcs/InputStreamSource_2>

Before a checkin or checkout operation to or from a store or location, verify that the FCS server defined in the FCS_URL on that store or location is running. If not, the operation will fail. Additionally, you should verify factors that may also affect the operation, like a LAN line being down.

The following diagram illustrates how a checkout operation works:

Checkout Sequence



The client makes a checkout request passing in a BusinessObjectProxy (BOP) object to the checkout(). The BOP contains the object ID, the file formats, and the names of the files to checkout. The ENOVIA Live Collaboration Server returns a ticket and the following is performed:

1. Fire Check trigger.
2. The file is replicated to the destination store/location, if needed.
3. Fire Override trigger.
4. Fire Action trigger.
5. Send a Ticket to FCS.

The FCS then sends the file to the requesting client.



Checkout Locations

ENOVIA Live Collaboration attempts to checkout the requested item as follows:

Use Locations that are part of the checkout person's preferred site. If none of these locations contain the newest copy, then:

Use any location that contains a valid copy of the file. This means that a "sync on demand" is performed - the requested file is copied to the user's preferred location, and then the local file checkout is performed.

Some checkouts allow the location to be overridden.



FCS Zip Support

FCS provides the ability for the client (the JSP page) to specify the download format, which includes the ability to compress the download stream (using Java zip support). This provides the following benefits:

- Multiple files can be downloaded in one archive stream (zip). This allows the current browser to deal with multiple files. (The browser can launch the appropriate zip program, such as WinZip).
- For a single file download, the compression, though optional, reduces the number of bytes that are transferred.

The checkout method of the checkout package allows a Boolean argument (`isZip`) to indicate use of the Java zip functionality.



Bulk Document Checkout

FCS also lets the client (the JSP page) download multiple files without compressing them into a single zip file. When you check out multiple files without compressing their contents, the files are checked out via a multi-part HTTP stream containing the uncompressed content of all files requested for check out.

Since multi-part HTTP streams are not supported by all browsers, you can only use this feature with FCS clients.

In the `doIt()` method of the `Checkout` class, the parameter `zipOutput` controls the checkout of multiple files.

- If the parameter `zipOutput=true`, the files are checked out as a single zip file.
- If the parameter `zipOutput=false`, the files are checked out as part of a multi-part HTTP stream.

It is your responsibility to correctly parse the multi-part download stream and break it up into multiple files.

For more information on the `zipOutput` parameter, see [Your Own Checkout](#) and [Using the com.matrixone.fcs.mcs](#).



Your Own Checkout

To write a client that performs a checkout, use the `doIt()` method of the `Checkout` class passing in various parameters. The `Checkout` class that you use depends on the kind of client you are writing. These APIs are:

- `com.matrixone.fcs.mcs`--Use this when the client performing the checkout is located on the ENOVIA Live Collaboration Server/FCS server.
- `com.matrixone.fcs.http.Http`--Use this when the client performing the checkout is a browser client.
- `com.matrixone.fcs.fcs`--Use this when the client performing the checkout is Java program.



Using the com.matrixone.fcs.mcs

The `com.matrixone.fcs.mcs.doIt()` method has four variants. All of the versions return a `TicketWrapper`. Use one of these two methods to do a simple file checkout where no manipulation of the checked out file(s) is required and if the file content is small. Use the second variant if a location override might be needed.

```
doIt(Context ctx, contextConnect, list) throws MatrixException  
doIt(Context ctx, contextConnect, locationOverride, list)  
throws MatrixException
```

Use one of these methods when you have to perform some additional file manipulation. For example, you want the checkout operation to create a zip file of the content. Use the second variant to allow a location override.

```
doIt(Context ctx, zipOutput, outFileName, contextConnect, list)  
throws MatrixException  
public static TicketWrapper doIt(Context ctx, zipOutput,  
outFileName, contextConnect, locationOverride, ArrayList list)  
throws MatrixException
```

The arguments include:

Argument	Description
Context	The ENOVIA Live Collaboration context.

<code>zipOutput</code>	Boolean. If the value is true, creates a zip file to hold the checked out files. If the value is false, checks out the files as part of a multi-part HTTP stream.
<code>outFileName</code>	The name of the zip file.
<code>contextConnect</code>	
<code>locationOverride</code>	The URL to the location where the checkout should be performed.
<code>list</code>	An array of files to checkout.



Using the com.matrixone.fcs.http Methods

The `com.matrixone.fcs.http. doIt()` method has four variants, however, two are deprecated. Of the remaining methods, one allows users to override the location of the checkout via the `locationOverride` parameter.

- To perform a simple checkout, without a location override, use this `doIt()` method:

```
doIt(Context ctx, String[] boids, String[] fileNames,
String[] formats, String[] locks, String[] paths, boolean
zipOutput, String outFileName, String errorPage,
HttpServletRequest req, HttpServletResponse res)
```

- To enable location overrides, use this `doIt()` method:

```
doIt(Context ctx, String[] boids, String[] fileNames, String[]
formats, String[] locks, String[] paths, boolean zipOutput,
String outFileName, String errorPage, String locationOverride,
HttpServletRequest req, HttpServletResponse res)
```

The following table describes the arguments that must be passed on a checkout operation:

Argument	Description
<code>Context</code>	The ENOVIA Live Collaboration context.
<code>boids</code>	An array of business object IDs containing the files to checkout.
<code>fileNames</code>	An array containing the names of the files to checkout.
<code>formats</code>	An array containing the formats of the files to checkout.
<code>locks</code>	An array containing the lock state of the files after the checkout is complete.
<code>paths</code>	An array containing the paths to the files
<code>zipOutput</code>	Boolean. If the value is true, creates a zip file to hold the checked out files. If the value is false, checks out the files as part of a multi-part HTTP stream.
<code>outFileName</code>	What the file is called locally after the checkout.
<code>errorPage</code>	The URL location to the error page to display if the checkout operation fails.
<code>locationOverride</code>	The location specification if you want to override the default location from which to checkout the file.
<code>req</code>	The HTTP request.
<code>res</code>	The HTTP response.

The following table describes what happens when you specify the `locationOverride` switch is specified:

Scenario	FCS Used
Store specified is a valid location	The FCS at the specified store is used
A Store is not specified but there is a valid location	The FCS at the specified store is used
A Store is specified and there is a valid location, but it does not have a valid FCS URL at the specified store	The store FCS is used
A Store is specified and there is a valid location but it does not have valid	The store FCS is used

FCS URL at the specified store	
Location specified is invalid	Store is used during checkout
Location specified is not in Store specified.	Store is used during checkout



Sample Checkout Code

The following example illustrates how to write a JSP that allows a location override.

```
.
.
.

<%
Context ctx = Framework.getFrameContext(session);
String errorPage = "/fcs/errorPage.jsp";
String[] boids = request.getParameterValues("boid");
String[] fileNames = request.getParameterValues("fileName");
String[] formats = request.getParameterValues("format");
String[] locks = request.getParameterValues("lock");
String[] paths = new String[fileNames.length];
boolean useZip = request.getParameter("useZip") != null;
String zipName = null;
if (useZip)
{
    zipName = request.getParameter("zipName");
}
TicketWrapper ticket = HttpCheckout.doIt(ctx, boids, fileNames,
formats, locks, paths, useZip, zipName, errorPage,
locationOverride, request, response);
String ticketStr = ticket.getExportString();
String ftaAction = ticket.getActionURL();
%>
<html>
<body>
    <form method="post" name="FcsForm" action="<%=>ftaAction%>">
<br>
<input name="<%=>McsBase.resolveFcsParam("jobTicket")%">
value="<%=>ticketStr%" size="90"><br>
<input name="<%=>McsBase.resolveFcsParam("failurePage")%">
value="<%=>Framework.getFullClientSideURL(request,response,error
Page)%>" size="90"><br>
<input name="<%=>McsBase.resolveFcsParam("attachment")%">
value="false" size="90"><br>
</form>
<script>
document.forms["FcsForm"].submit();
</script>
</body>
</html>
```



Using the com.matrixone.fcs.fcs Methods

The com.matrixone.fcs.fcs package has a single doIt() method with this signature:

```
com.matrixone.fcs.common.JobReceipt doIt(FcssServlet servlet,
FcsContext fcsContext, javax.servlet.http.HttpServletRequest req,
javax.servlet.http.HttpServletResponse resp)
```

Replication

By default, FCS file replication is enabled to save on file replication resources over your network. Replication is necessary when there is more than one file store and thus multiple FCSs. This configuration allows you to have FCSs closer to the user base.

The following topics are discussed:

- [About Replication](#)
- [Replication from a Specific Source and Destination Location](#)
- [Replication Through Stores](#)

About Replication

Sites and locations enhance checkin, checkout, and open for view or edit performance for clients that require WAN access to a centralized storage location. The central store (default) is mirrored to one or more remote machines (locations). In this way, a client at the remote site has LAN access to the data.

When a user performs a file checkin, the system writes the file to that user's preferred location. If it is not specified, it writes the file to the store's default path. In the schema, the business object is marked as having a file checked in at this location. When another user requests the file for checkout, the system attempts the checkout from these locations:

Locations that are part of the checkout person's preferred site. If none of these locations contain the newest copy, then;

The store's default location. If this does not contain the newest copy, then continues to 3.

Any location that contains a valid copy of the file. This means that the system performs a sync-on-demand--the system copies the requested file to the user's preferred location then performs the local file checkout.

For example, if you are in the United States and you have an office in Boston, and another in Paris you could have two FCSs: one in each office. If replication is used, it does not matter where the checkout occurs because the FCS ensures that both locations are synched up and ensures that duplicates are removed.

A user can only have one site. A site can have one or more locations. A functional store contains one or more locations. For example:

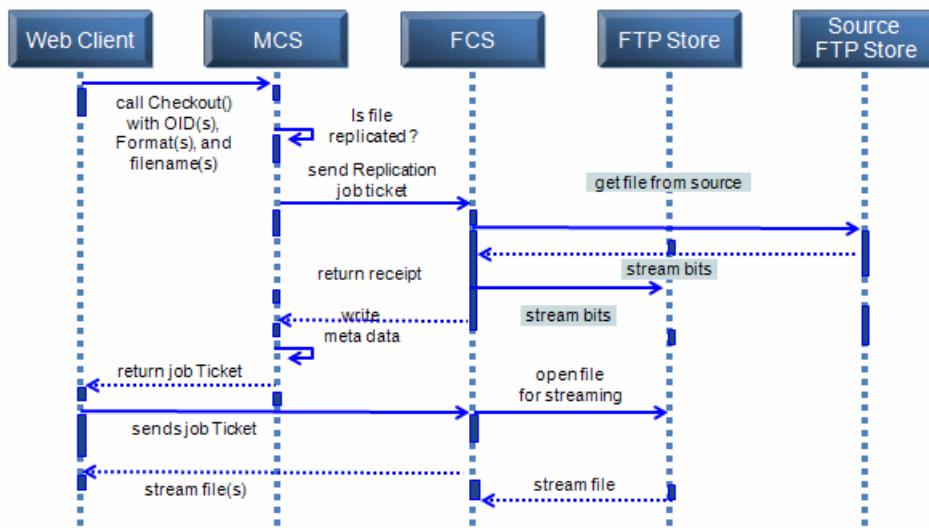
- The user is located in Boston, Site = Boston.
- There are 2 possible Locations: Boston and Paris.
- There are 3 possible functional Stores: Marketing, Sales, Engineering

For example, if a user is an engineer who works in Boston. The site would be Boston, the Store could be engineering and the location could be Paris or Boston. To switch the default location, use the locationOverride switch.

When you need to replicate multiple files, the files are replicated in a single batch rather than one at a time to streamline the replication process. The system determines the list of files to be replicated and then groups them by "destination location." Files in the list can have different destination locations as long as they all have the same FCS URL since the FCS URL for the first group of files is used for the checkout ticket. The system then replicates each group of files to the destination location in a single batch and commits the metadata. If a replication process fails, subsequent items will not be committed but all previously committed items will remain present in the database.

A checkout operation with replication looks like this:

Checkout with Replication Sequence



If a transaction fails the ENOVIA Live Collaboration Server goes back and deletes the files on the remote server. If there are duplicates, the ENOVIA Live Collaboration Server deletes the duplicates that are on the FTP server that is logically part of the FCS server.

If there is more than one FCS server, the ENOVIA Live Collaboration Server cleans up the copy of the file where it was last checked in. All other copies are marked obsolete. In addition, you can write a cron job to do the tidystore cleanup operation.

Replication from a Specific Source and Destination Location

With FCS file replication, the source location is the *first* location that has the current file. The destination location is the requesting client's preferred location. However, you can execute a custom replication program to override this default behavior. This program must be defined on the store and any location involved in replication with this store will use this program. Defining a program on a store allows for greater control over when the program is executed. The custom program must implement the following Java interface:

```
Package com.matrixone.fcs.mcs;
public interface ReplicationTrigger {
    /**
     * Called prior to replication.
     * @param cntx
     * @param inList - list of @see {@link BusinessObjectProxy}
     * that represent the source for file that need replication
     * @param copyList - list of @see {@link BusinessObjectProxy}
     * that represent the destination for file
     * that need replication
     * @return replacement inList, can be the original inList
     * @throws Exception
     */
    public ArrayList check(Context cntx,
                          ArrayList inList,
                          ArrayList copyList) throws Exception;
}
```

To define the custom replication program on the store, execute the following in MQL:

```
modify store NAME trigger CustomReplicationProgram;
```

The MX_FCS_REPLICATE_PROGRAM_NAME global variable should no longer be used. This program operates on one file at a time and is no longer needed now that files are replicated in a single batch. If you are currently using this program, you must replace it with the custom replication program described above. The system will now issue a warning if this legacy program is still in use.

Replication Through Stores

In certain cases it may be beneficial to replicate files from a location through a central store. To enable this functionality, configure the following class on the store in MQL:

```
modify store NAME trigger com.matrixone.fcs.backend.ReplicateThroughStore;
```

Business Object Proxies (BOPs)

This section describes business object proxies used when copying files.

The following topics are discussed:

- [About Business Object Proxies](#)
- [File Copy](#)
- [Copy Locations](#)

About Business Object Proxies

BusinessObject is the base class for all business objects. BusinessObjects represent data, such as a Part or a Document and AdminTypes defined the meta-data. For a part there would be an admin object, of type "type" called Part. The Part BusinessObject is an instance of type Part. BusinessObjects can contain one or more formats. Each format can have one or more files associated with it. The filename must be unique within the format.

The Business Object Proxy represents one file within a BusinessObject. The BOP contains an object id, a format name, and a filename. It also contains several Boolean flags indicating the state to leave the object when the file operation completes. For example, one flag indicates that the BusinessObject be locked after the checkout operation is completed. Another flag says to append a new file to the BusinessObject (unless set, all previous files within the object, regardless of format, are removed from the business object during checkin)

One business object proxy can have multiple files that are unique within a given format. Formats are metadata policy lists.



File Copy

The file copy operation uses a method that performs the file copy from the original BusinessObject, and then checks the meta data into the copy BusinessObject. For example:

```
BusinessObjectProxy inBop = new  
BusinessObjectProxy(oid,format,filename,false,false)  
  
BusinessObjectProxy copyBop = new  
BusinessObjectProxy(oid2,format2,filename,append,false)  
  
ArrayList inList = new ArrayList();  
inList.add(inBop);  
copyList.add(copyBop);  
com.matrixone.fcs.mcs.Copy(context,inList,copyList);
```

The file name in `inBop` is copied to the `copyBop` object. As a result, there are two files in the store. There is no connection between the two files or business objects.



Copy Locations

It is assumed that both the source and destination locations will be computed based on users' site preferences. If you do not specify the destination location, the source location is used. The FCS specified by the destination location is used to do the actual file movement.

FcsClient

The FcsClient is the client side library that you use to implement the client side of the FCS.

The following topics are discussed:

- [InputStreamSource_3 Interface](#)
- [Getting and Setting the Correlation ID](#)
- [Debugging with the Correlation ID](#)
- [Uncompressing the Data Stream](#)

InputStreamSource_3 Interface

FcsClient users must implement the InputStreamSource_3 interface. This interface follows an "Iterator" pattern by accessing files from a list one at a time. The FcsClient calls next() to indicate that the next file is being operated on, and then calls reset() to move the current file pointer to the beginning of the list.

The InputStreamSource_3 interface is an enhanced version of the InputStreamSource_2 interface.

InputStreamSource_2 Interface:

```
package com.matrixone.client.fcs;
import java.io.IOException;
import java.io.InputStream;

public interface InputStreamSource_2 {
    /**
     * get the next item to be processed
     */
    public InputStream getInputStream() throws IOException;

    /**
     * get the next file name to be processed
     */
    public String getFileName();
    public boolean hasNext();
    public boolean hasNext();
    public String getFormat();

    /**
     * @return
     */
    public long getFileSize();
    public void next();
    public void reset();
}
```

InputStreamSource_3 includes an extra method called getCorelationID(), which adds a fourth option to the list of ways in which to pair items in the Business Object Proxy (BOP) list and the actual files, using the correlation ID as the key.

- Ordered (that is, 1st to 1st, 2nd to 2nd, etc.)
- By unique filenames
- By unique filenames and format
- By correlation ID

InputStreamSource_3 Interface:

```
package com.matrixone.client.fcs;
import java.io.IOException;
import java.io.InputStream;

public interface InputStreamSource_3 extends
InputStreamSource_2 {
    public String getCorelationID();
}
```

The InputStreamSource_3 interface is easier to use than the InputStreamSource_2 interface. The latter is still available, but is recommended not to be used. The original InputStreamSource interface is no longer available and should not be used.



Getting and Setting the Correlation ID

You should implement InputStreamSource_3 such that the getCorelationID() method returns the corresponding ID as the BOP.

```
public interface InputStreamSource_3 extends InputStreamSource_2 {
    public String getCorelationID();
}
```

You should invoke bproxy.setCorelationID(*corelationId*) for each BOP on the list.

```
public class BusinessObjectProxy implements Cloneable, Serializable {
    public void setCorelationID(String corelationId) {
        _corelationId = corelationId;
    }
}
```

}

For convenience, a unique ID generation method has been added to the FcsClient. Applications can use this method to generate a unique correlation ID.

```
public class FcsClient {  
    public static String generateCorelationId() {  
        return System.currentTimeMillis() + rand.nextInt() + "";  
    }  
}
```

Debugging with the Correlation ID

For debugging purposes, [FcsSubmit](#) accepts a `-correlateWithId` argument, which causes the checkin to be run in correlationID mode. For example:

```
String fcsargs[] = {"-user", "tester1",
                     "-password", "",
                     "-type", "FCSTstType1",
                     "-name", "TSTBO",
                     "-rev", "A",
                     "-url", _fcsurl,
                     "-policy", "FCSTstPolicy1",
                     "-vault", "eService Production",
                     "-format", "FCSTstFormat1",
                     "-store", "TESTSTORE",
                     "-count", "+_COUNT,
                     "-donotexit",
                     "-correlateWithId"};
FcsSubmit.main(fcsargs);
```

Uncompressing the Data Stream

By default, the FcsClient receives a compressed data stream when checking out more than one file. The compressed data stream conserves network bandwidth but negatively impacts the transfer rate between the FCS and FcsClient. When you check out without compression, the transfer rate between the FCS and FcsClient improves and FCS CPU consumption is minimized.

You can choose to check out with compression when you need to conserve bandwidth, or without compression when bandwidth is available. You can uncompress the data stream in either of two ways:

- Set the `zipOutput` parameter to false when retrieving the checkout ticket, using the `dolt` method in the `com.matrixone.fcs.mcs.Checkout` class:

```
/**  
 * @param ctx  
 * @param zipOutput  
 * @param outFileName  
 * @param contextConnect  
 * @param list  
 * @return  
 * @throws MatrixException  
 */  
public static TicketWrapper doIt(Context ctx, boolean zipOutput, String outFileName,  
String contextConnect, ArrayList list) throws MatrixException
```

You can choose to ignore the `outFileName` parameter or set it to null. The `locationOverride` parameter needs to be used only if you need to override the FcsClient's preferred location. The `contextConnect` parameter is the MCS URL. The `list` parameter contains the list of BOPs involved in the checkout operation.

The checkout operation is performed as usual. The output stream compression switch is in the checkout ticket returned by the doIt method.

- Enable the checkout method for the FcsClient to unzip the zip stream. A boolean unzipFiles parameter is available in the FcsClient.checkout method for this functionality:

Large Files

This section describes how to handle large files when using FCS.

The following topics are discussed:

- [Requirements](#)
- [Checkin](#)
- [Checkout](#)
- [FcsClientLargeFile Class](#)
- [Hard Link for Local Clone](#)

Requirements

For HTML/JSP-based applications, including the ENOVIA product suites and custom programs, large file checkins require both of the following:

- Checkins must be targeted for a FCS-enabled store/location.
- Checkins must be invoked via the configurable file upload applet (see *Live Collaboration Administrator's Guide* for setup information.)

It is recommended that you enable both FCS and the file upload applet for all implementations, even if you do not foresee working with files larger than 2 Gbs.

For files larger than 2.5 Gb, you should write your own checkout JSPs.

The file transfer mechanism for FCS checkins and checkouts uses Transfer-Encoding=chunked per the HTTP 1.1 specification, which is supported on all ENOVIA Live Collaboration supported application servers. However, there are certain third-party vendor components that may not support this protocol correctly. These limitations are described below.

Checkin

If there is a web server located between the client (either Matrix Navigator or ENOVIA products) and the FCS, some web servers will not pass the file-transfer request through properly.

Apache, SunOne, Websphere, and the WebLogic plugin encountered errors. Typical errors (in the web server's log) may report that chunking is not supported, content-length is not set, or there is a badly formed POST body. These may be issued by the web server itself, or by web server plugins that pass the request on to an Application Server.

STATUS/WORKAROUND: This is not an issue for files < 1.9 gigabytes (2040109465 bytes) which do not use the chunked protocol. If you do not typically handle files larger than this, there is no problem. However, if you frequently handle files larger than this, it is necessary to define FCS URL's so they do NOT go through a web server/plugin that does not support this aspect of HTTP 1.1.

There is a delay at the end of a checkin for files < 2 Gb after the server returns a receipt to the client. If ematrix.ftp.debug=true, you can see that the receipt was output to the server, but there is a several second delay before the operation completes and the progress dialog goes away.

Checkout

Downloading a file bigger than 4 Gb should not be handled by Internet Explorer by default.

For files less than 4 gigabytes, no workaround is needed (although the workaround described here is for files over 2 gb). If you need to download larger files, you will need to configure the client-side applet (FileDownloadApplet) to be used instead of emxCommonDocumentCheckout.jsp for use with the applications, by setting the following system property to true in emxsystem.properties. The default is false.

`emxFramework.UseDownloadApplet = true`

When the download applet is enabled, it is used only for downloading files over 2 gb. When it is used, the files are saved to the file system; however they cannot be opened via the applet without making changes to the registry for the I.E. client. The following two links explain the problem and note some registry settings that can be modified to override this behavior:

<http://windowsxp.mvps.org/ie/elevlocalfile.htm>

<http://www.all-usernet-archive.com/File.asp?service=15823>

For Fire Fox, you can set security.checkloaduri to false (in `about:config` URL) to allow large files to be opened.

Alternatively, you can open the files outside of the application via the file system.

If you frequently handle files larger than 2 gigabytes, configure your FCS URLs to directly point to the FCS application server, with no intermediate web server. Also configure both the FileUploadApplet and the FileDownloadApplet to replace the corresponding checkout/checkin JSP pages in the applications.



FcsClientLargeFile Class

The FcsClientLargeFile class can also be used to handle large file checkin and checkout operations. The class contains one checkin method and two checkout methods. These methods improve large file checkin and checkout performance by using hard links to avoid file copy operations and by using an optimized procedure for file copy operations rather than a high-level transport protocol such as HTTP.

Hard links are directory references or pointers to a file on a storage volume. Hard links can only be created on the same file system where the file is stored. The following three directories are used by the checkin and checkout methods. These directories must be on the same file system for hard links to work.

- **Application Working Directory** - is used by the FCS client application to read or write files. This directory is managed by the application and can only be accessed by the application. The FCS does not have access to this directory.
- **Stage Directory** - is used to exchange files between the Application Working directory and the Store/Location directory. This directory manages file permission changes and can be accessed by both the application and the FCS. This directory is managed by the application.

Files and hard links are not created in this directory. They are created in the temporary folders created by the command in this directory.

- **Store/Location Directory** - is used by the FCS to store files. This directory is managed by the FCS and can only be accessed by the FCS. The application does not have access to this directory.

In UNIX environments, application and FCS users must have access to the "ln" and "chmod" commands. In Windows environments, hard links are created through the CreateHardLink executable program.

Before you can use the FcsClientLargeFile class, the FCS must have the appropriate access to the file system. File access should be carefully managed for each directory. For information about the FcsClientLargeFile class, see the *Studio Customization Toolkit Reference Guide (Java Docs)* that are installed with the Studio Customization Toolkit.

Setting Up File Access

The following access should be given to each directory:

On UNIX:

- Application Working Directory - application user should have full control: drwx-----
- Stage Directory - everyone should have write and execute access: d-wx-wx-wx
- Store/Location Directory - FCS user should have full control: drwx-----

On Windows:

From the Properties window for each directory, access the Security tab and grant the following access:

- Application Working Directory - application user should have full control
- Stage Directory - application and FCS users should have full control. A group can be created for application users to make this task easier.
- Store/Location Directory - FCS user should have full control.



Hard Link for Local Clone

The hard link for local clone feature improves the efficiency of the FCS local clone service by creating hard links on copied files. This saves storage space and is particularly important for large file support. MQL commands are provided for evaluating the size of a store or location using ENOVIA V6 file metadata in order to assist System Administrators in planning replication and archiving processes.

The hard link for local clone feature improves the performance of FCS clone/copy services in the special case of local clone. By "local" is meant that the source (copied) file and destination (copy) file are managed by the same FCS. The difference between the regular FCS clone/copy and FCS hard link clone/copy is the file-system type of the files created: the former creates files using buffered, bit-wise copy while that latter creates links. Creation of a hard link is possible only when the file that is being copied and its copy belong to the same file system. As the FCS server cannot check for this condition, a regular file copy is performed whenever hard link creation fails.

Hard link creation is not always faster than copying a file, because the performance of hard link creation (from Java code) is both OS and hardware dependent. A hard link threshold property is, therefore, provided to allow setting of the maximum file size for which copying is faster than hard link creation. In other words, copying of a file that is smaller than the threshold value is faster than creating a hard link for that file. You can override the default, OS-dependent threshold values provided in the FCS code, if desired, by setting the `ematrix.fcs.hardlink.threshold` property in the `framework.properties` file, which is located in the `SERVER_INSTALL\managed\properties\` (Windows) or `SERVERHOME/managed/properties/` (UNIX) directory. The value of this property is a file size in bytes: all files smaller than this size are cloned using file copy, files larger than this size are cloned by hard link copy. To disable the hard link for local clone feature altogether, you can set the `ematrix.fcs.hardlink.threshold` property to -1. This may be useful when using a new prefix for a store, assuming that the new prefix path and the former store path are on different file systems.

Introduction of a hard link in store and location directories creates a gap between the store/location logical size, which is computed from ENOVIA V6 metadata, and its file system size. To handle this, MQL commands are provided to retrieve

store/location logical size. These commands improve store and location administration tasks, such as store archiving and replication.

In addition, **print store/location XXX select size.*** commands are provided to compute the total size of the files physically located in a store/location path by using ENOVIA V6 file metadata. Obsolete files are also included in the computation. For example, if you check a file into store A, the size of the store returned by the **print store A select size.byte** command is increased by the byte size of this file. If you check a file into location locA and replicate store A, the size of store A does not change but that of locA does.

On UNIX systems, evaluating store XXX size provides the same result as the shell command **du -l /store_XXX_path/**. The **-l** option of the **du** command counts file size as many times as the file is hard linked.

The store/location selectable is **size**, which is associated with a **unit** sub-selectable (byte, mb, or gb), which in turn specifies the size unit. For example, the command **print store/location XXX select size.byte** returns a size in bytes. In like manner, the argument **select size.mb** returns a size in megabytes, while **select size.gb** returns a size in gigabytes. The **print store/location XXX select size.[byte|mb|gb]** command does not create persistent data (hard links).

The files of a hashed store that is managed by FCS and those recorded by the MCS metadata (businessobject files) have different names. However, there is a one-to-one mapping between metadata files and store files. The life cycle of persistent data (hard links) through FCS operations is as follows:

- A client application checks out a file that is the result of a hard link local clone/copy. In this case, FCS reads and returns file data as with any other file.
- A client application updates a file that has been cloned one or more times with hard link for local clone. In this case, FCS creates a new file in the store folders (with encoded/hashed names) during update checkin, and then deletes the former version of the file. However, as the file is a hard link, it is not removed from the file system. The file system operation performed is in fact a (POSIX) unlink operation.
- A client application updates a file that is the result of a hard link local clone/copy. In this case, FCS creates a new file in the store folders (with encoded/hashed names) during update checkin, and then deletes the former version of the file. However, as the file is a hard link, it is not removed from the system. The delete operation performed is in fact a (POSIX) unlink operation.
- A client application deletes a file that has been cloned one or more times with hard link local clone. In this case, FCS deletes the corresponding encoded/hashed file in the store folder. As this file is the target of one or more hard links, this delete operation is in fact a (POSIX) unlink operation. So again, the file is not removed from the file system.
- A client application deletes a file that is the result of hard link local clone/copy. In this case, FCS deletes the corresponding encoded/hashed file in the store folder. As this file is a hard link, this delete is in fact a (POSIX) unlink operation. So once again, the file is not removed from the file system unless it is the last file link.

The following are some limitations of the hard link for local clone feature:

- The hard link for local clone feature works only for captured, hashed stores.
- The hard link for local clone feature works only for stores that have a non-void FCS URL.
- If you create a store prefix that is the mount point of another file system or a junction point of another volume (NTFS), then the hard link for local clone feature should be disabled for the FCS managing this store (by setting `ematrix.fcs.hardlink.threshold=-1`) in the `framework.properties` file.
- When performing store or location synchronization, the hard links created in the store or location path are not reported in the sync destination path. So, the volume of this synchronization operation may exceed the file system size of the source store or location. To correctly plan your synchronization operation, you must issue a **sync store** command together with the **validate size** option.
- The system rich client application has an archive store/location function, which checks out files before archiving them. Archive files created by this function do not contain hard links and are, therefore, larger than the store/location file system size.

File Stream Filters

You may want to configure a Java class in a FCS Store/Location to perform processing on the file stream that is being written to the location (for checkin) or read from the location (checkout). This section explains how. You will need to write a Java class that is the filter, a params file, and name the location. Filters you implement must operate on the streams in an efficient manner. For example, the filter should operate on the stream whenever possible, and not write the files to disk as an intermediate step. Overall FCS performance will suffer, taking away from the FCS's value proposition, if not implemented correctly.

In this section:

- [Java Class to Filter File Stream](#)
- [Sample Filter](#)
- [The Parameters File](#)
- [Filter Calls](#)

Java Class to Filter File Stream

ENOVIA does not ship any filters. You need to write your own filter as a Java class. This java class will need to implement methods to allow it to participate in the streaming of bytes to/from the underlying storage.

The filter is defined with the store meta data. A filter that is defined in the store will be applied at the locations. Since locations are added to stores, this data then propagates.

The file size of a checked in file will represent the number of bytes streamed in from the client when the file was checked in. Therefore, your the filter should not modify the file in a way that changes its size or the file size reported by 'print bus select format.file.size' will be different from the number of actual bytes stored on disk.

The FCSStreamFilter is the Java interface that must be implemented and deployed on the FCS Server. The filter implementation needs to return the streams that are requested.

```
Package com.matrixone.fcs.fcs;
import java.io.InputStream;
import java.io.OutputStream;
import com.matrixone.fcs.common.FcsException;
import com.matrixone.jdom.Element;
public interface FCSStreamFilter {
    /**
     * Called when the fcs is doing a checkout.
     * The implementation needs to use the underlying stream to
     * get the actual data
     *
     * @param fName - file name - hash name
     * @param info = the location info.
     * @param item - the item data structure
     * @param underlyingStream - an opened stream to the
     * underlying data
     * @return modified stream
     * @throws Exception
     */
    public InputStream getCheckoutStream(Item item, LocationInfo
info, FcsContext context, InputStream underlyingStream)
        throws Exception;

    /**
     * Called when the fcs is doing a checkin. The
     * implementation can either act as an end point or a filter.
     * The implementation needs to use the underlying stream to
     * get the actual data
     *
     * @param item
     * @param info
     * @param context
     * @param underlyingStream
     * @return
     * @throws FcsException
     */
    public OutputStream getCheckinStream(Item item, LocationInfo
info, FcsContext context, OutputStream underlyingStream)
        throws FcsException;

    /**
     * Called with params for the location, if any. The params
     * @param params Root of the filter params
     */
    public void init(Element params);
}
```

Sample Filter

The following code illustrates the most basic filter, it does not actually do anything, but illustrates the structure of the filter approach.

The sections in bold represent the an appropriate way to process the output stream as it is begin written by the FCS server to the original stream.

```
public class TestFilter implements FCSStreamFilter {
    private Element _params;
    class MyFilteredOutputStream extends FilterOutputStream
    {
        private int _bytesWritten;
        public MyFilteredOutputStream(OutputStream out) {
            super(out);
        }
        public void write(byte[] b, int off, int len) throws
IOException {
            super.write(b, off, len);
            _bytesWritten+= len;
        }
    }
    public OutputStream getCheckinStream(Item item, LocationInfo
info, FcsContext context, OutputStream underlyingStream) throws
FcsException {
        return new MyFilteredOutputStream(underlyingStream);
    }
    public InputStream getCheckoutStream(Item item, LocationInfo
info, FcsContext context, InputStream underlyingStream) throws
Exception {
        return underlyingStream;
    }
    public void init(Element params) {
        _params = params;
    }
}
```

The Parameters File

The parameters file provides an ability to specify parameters that are passed into the filter code in the init method.

The `filterParams` section is passed in as a JDOM Element.

Stores that are specified as either 'filter' or 'external' can have additional parameters provided in a param file. However params are not mandatory.

`<filterParams>` must be included to identify those params which are to be passed to the filter class.

`<externalParams>` must be included to identify those params which are to be passed to an external URL.

Here is a sample params file:

```
<?xml version="1.0" encoding="UTF-8"?>
<storeParams>
    <externalParams>
        <checkinParams>
            <param name="param2">value2</param>
        </checkinParams>
        <checkoutParams>
            <param name="param1">value1</param>
        </checkoutParams>
    </externalParams>
    <filterParams>
        <param name="param1">value2</param>
        <param name="param2">value2</param>
    </filterParams>
</storeParams>
```

Filter Calls

The filter clause of the MQL Add Store statement (as well as Modify Store and Add/Modify Location) lets you add a Java class to allow filtering or special processing on the data stream during file check in and check out.

The keyword filter indicates the store has a Java class configured to perform filtering. You can use either the file or the params keywords to provide the parameters as necessary.

```
add store NAME [filter FILTERCLASS [| file PARAMSFILE | |]
| params PARAMSSTRING| ;
```

FILTERCLASS is the name of the Java class that does filtering on the input or output stream. This class must implement the com.matrixone.fcs.fcs.FCSStreamFilter interface.

PARAMSFILE is the name of an XML file containing parameters to pass to the filtering class. Refer to [The Parameters File](#) for details.

PARAMSSTRING is a string of XML code containing parameters to pass to the filtering class, as an alternative to using an external PARAMSFILE.

When adding or modifying external stores, you can combine this parameters file with the one that is specified for the Web Service (external) store.

For example:.

```
add store test type captured directory c:\storesDir\ filter
filterClass file paramsFile.xml
```

The system verifies the validity of this class at the time the store is configured, thus, the class must be available in the classpath at the time of schema creation.

Debugging

This section describes using FCS trace and FCSTools for debugging.

In this section:

-  [FCS Trace](#)
-  [FCSTools](#)

FCS Trace

The server diagnostic trace tool is very helpful in debugging processes. The trace tool allows FCS trace information to be sent to a file or to standard output. When FCS trace is enabled on the MCS, both the MCS and FCS processing the request will have tracing enabled.

The ematrix.debug=true tool can also be used to enable debug.

Tracing for FCS can be turned on (and off) using the following MQL command:

```
trace type fcs |filename FILENAME;
                |on
                |off
                |text STRING
```

The following topics are discussed:

- [File Clause](#)
- [On Clause](#)
- [Off Clause](#)

File Clause

Specifying a file with the `file FILENAME` clause turns the specified type of tracing on, and redirects the output to the specified file. There is no need to use the keyword "on". If a filename has already been specified for another type of tracing within the current session, that filename will be used, and the one specified here will be ignored, but the tracing will be enabled.

You can pre-configure the FCS to redirect it's output to a log file instead of the standard server stdout by setting `ematrix.fcsLogFile` to point to that log file. This parameter, like all FCS parameters can be set in either `web.xml` or as a `-D` parameter passed to the server startup process.

If the filename specified for any tracing already exists in the directory `MX_TRACE_FILE_PATH`, that existing file will be copied to a backup whose name is constructed by prepending a time-specific prefix to the filename, in the form `"yyyymmddhhmmss__FILENAME."`

On Clause

The `on` modifier will turn the specified tracing on and send the trace information to `stdout`, unless a trace file is already in use by another type of tracing.

Off Clause

The `off` clause turns the specified type of tracing off. If other tracing types have been turned on, they will stay on. Tracing that is turned on via .ini file settings can only be turned off using the keyword `all`. For example:

```
trace type fcs off;
trace type all off;
```

When all types of tracing directed at the same file are turned off, the file is closed. In order to resume tracing to a file, any subsequent command must specify a filename. If it doesn't, tracing will be resumed with `stdout` as the destination.

FCSTools

The FcsReceive and FcsSubmit tools allow you to test the FCS. These tools have lots of arguments and these arguments change often. The usage is subject to change so it is not documented here. Running these tools without any arguments produces the usage.

This is how you run, for example, FcsReceive:

```
set MX_HOME=d:\ematrix  
java -cp %MX_HOME%\java\lib\eMatrixSevletRMI.jar;%MX_HOME%\java\lib\mx_jdom_1.0.jar  
com.matrixone.fcs.tools.FcsReceive
```

Running FcsSubmit is similar.

These tools act as both an ENOVIA Live Collaboration Server client, using the Studio Customization Toolkit, and the FCS client, using the FcsClient API. For more information on the Studio Customization Toolkit, see the *Programming Guide*. The [FcsClient](#) is the client-side library that you use to implement the client side of the FCS.

Note: Neither FcsReceive nor FcsSubmit support a Single Sign On (SSO) environment.

Custom JSP Pages for FCS

When the FCS is used with custom JSP pages, JSP pages must follow a specific coordination of steps to achieve file checkin and checkout operations.

In this section:

- ❑ [FCS with Custom JSPs](#)
- ❑ [General Pages common.inc](#)
- ❑ [Checkin Pages](#)
- ❑ [Checkout Page](#)
- ❑ [Error Page](#)

FCS with Custom JSPs

Using FCS with custom JSP pages requires specific steps to enable the flow of checkin/checkout operations.

Custom JSP pages must include several HTTP parameters in order to be FCS enabled. All HTTP parameters must be wrapped in a special function prior to being set. In addition, the checkin form allows page designers to specify any other additional parameters, which should not be wrapped in the special function.

For example, the page sent to the FCS would have these sections:

```
<form name...>
<!--FCS aware parameters-->
<input name="<%={McsBase.resolveFcsParam("failurePage")%}" value="<%={errorPage%}" size="90"><br>
<input name="<%={McsBase.resolveFcsParam("jobTicket")%}" value="" size="90"><br>
<!--user parameters that are optional -->
<input name="param_1">
<select name="select_1">
<option value="a">
</select>
</form>
```

The section commented as "FCS aware parameters" contains the special function that is required for all custom JSPs to be FCS enabled. The section commented as "User parameters" contains optional parameters outside of the special function.

Following are additional considerations for making JSP pages FCS enabled:

- All JSP pages must have a `jobTicket` parameter, which must proceed any "input type file" fields in the checkin pages.
- All JSP pages must have a `failurePage` parameter, which allows specification of the error page. The Job ticket typically contains the same error page; however, if FCS cannot read the job ticket, the `failurePage` parameter is used.
- For checkin pages, since tickets can be processed only once, create your applications to use single-click checkin. Double-click checkin will cause an exception.
- For checkout operations only, in addition to the above parameters, the `attachment` parameter is examined during checkout. If `attachment` is set to "true", the file contents are sent as an attachment, and the browser displays the "Save As" dialog box.

General Pages common.inc

This sets the character encoding for the response to send to the client.

```
<%@page contentType="text/html; charset=UTF-8"%>
```

The following topics are discussed:

- [index.jsp](#)
- [objSetup.inc](#)

index.jsp

This page sets up the business object and the context for all subsequent pages:

```
<%@ page
import="com.matrixone.servlet.* ,matrix.db.* ,java.util.* "%>
<%@ include file="objSetup.inc"%>
<%=context.getSession().getSessionId()%><br>
<%
String oid = obj.getObjectId();
Framework.setContext(session,context);
String each = "&boid=" + oid +
"&lock=false&format=generic&fileName=";
String line = "";
Enumeration formatList = obj.getFormats(context).elements();
matrix.db.File mxFile;
Format format;
while (formatList.hasMoreElements())
{
    format = (Format)formatList.nextElement();
    Enumeration fileList =
obj.getFiles(context,format.getName()).elements();
    while (fileList.hasMoreElements())
    {
        mxFile = (matrix.db.File)fileList.nextElement();
        line += "&lock=false&boid=" + oid + "&format=" +
mxFile.getFormat() + "&fileName=" + mxFile.getName();
    }
}
%>
<html>
<body>
<table BORDER=0>
<tr>
<td>2 part checkin Checkin (Checkin start/Checkin End)</
td><td>&nbsp;</td>
<td><a href="<%="response.encodeURL("twoPartCheckin.jsp?boid=" +
oid)%>">Go</a></td>
</tr>
<tr>
<td>1 part checkin Checkin (Pre-checkin/Checkin)</
td><td>&nbsp;</td>
<td><a href="onePartCheckin.jsp">Go</a></td>
</tr>
<tr>
<td>Checkout</td><td>&nbsp;</td>
<td><a href="<%="response.encodeURL("checkout.jsp?" +
ServletUtil.encodeURL(line ))%>">Go</a></td>
</tr>
</table>
</body>
</html>
```



objSetup.inc

This file is used to set up the testing of a specific business object, policy, attribute, etc.

```
<%
String rev = "A";
String vault = "eService Sample";
String policy = "test";
String type = "test";
String name = "test";
Context context = new Context("", "localhost");
context.setUser("creator");
context.setPassword("");
context.setVault("");
context.setLanguage("en-us");
context.setTimezone("EST");
context.connect(true);
BusinessObject obj = new BusinessObject(type,
                                         name,
                                         rev,
                                         vault);

if (!obj.exists(context))
{
    obj.create(context,policy);
}
else
{
    obj.open(context);
}
%>
```

Checkin Pages

The checkin operation can be performed in one of 2 ways.

- One-part checkin is designed to allow the IO operation to happen without already having a business object. For details about the one-part checkin process, see [One-Part Checkin](#).
- Two-part checkin is designed to have the business object present prior to the checkin operation. For details about the two-part checkin process, see [Two-Part Checkins](#).

Both checkin types rely on having two pages in a frameset. The pages are named *FCSForm and *MCSForm. The FCS form is used to actually submit the file data to the FCS. The ENOVIA Live Collaboration Server form is used to get the ticket from the ENOVIA Live Collaboration Server. Both forms use JavaScript to fetch ticket data and stuffing into the FCS form.

The following topics are discussed:

- [onePartCheckin.jsp](#)
- [onePartCheckinFCSForm](#)
- [onePartCheckinMCSForm.jsp](#)
- [preCheckinMCS.jsp](#)
- [checkinMCS.jsp](#)
- [twoPartCheckin.jsp](#)
- [twoPartCheckinFCSForm](#)
- [twoPartCheckinMCSForm.jsp](#)
- [checkinMCSStart.jsp](#)
- [checkinMCSEnd.jsp](#)
- [thankYou.jsp](#)
- [thankYouText.jsp](#)

onePartCheckin.jsp

This file holds the frameset used to set up the one-part checkin.

```
<%@include file="common.inc"%>
<frameset rows="30%,*" framespacing="0" frameborder="no"
border="1">
    <frame name="mcsForm" noresize marginheight="10"
marginwidth="10" border="1" scrolling="yes"
src="<%="response.encodeURL("onePartCheckinMCSForm.jsp?boid=" +
request.getParameter("boid"))%>" frameborder="0">
    <frame name="fcsForm" noresize marginheight="10"
marginwidth="10" border="1" scrolling="yes"
src="onePartCheckinFCSForm.jsp" frameborder="0">
</frameset>
```



onePartCheckinFCSForm

This file is the user visible form. The user uses this form to select one or more files for upload. One-part checkin only needs the file count to issue the ticket.

```
<%@include file="common.inc"%>
<%@ page
import="com.matrixone.servlet.Framework,com.matrixone.fcs.mcs.M
csBase" %>
<%
String errorPage =
Framework.getFullClientSideURL(request,response,"/fcs/
errorPage.jsp");
// number of potential files to upload
int numPotentialFiles = 5;
%>
<html>
<body>
<script>
function doFileUpload()
{
    var mcsForm = parent.frames["mcsForm"].document.forms[0];
    var actualFileCount = 0;
    <%
        for (int i = 0; i < numPotentialFiles;i++)
    {
```

```

%>
    if (document.forms[ "FCSForm" ].file_<%=i%>.value != " ")
    {
        actualFileCount++;
    }
<%
}
%>
mcsForm.numFiles.value = actualFileCount;
mcsForm.submit();
}

</script>
<form enctype="multipart/form-data" method="post"
name="FCSForm" action="">
<br>
Error Page: <input
name="<%>McsBase.resolveFcsParam( "failurePage" )%>" value="<%>errorPage%>" size="90"><br>
JobTicket: <input
name="<%>McsBase.resolveFcsParam( "jobTicket" )%>" value=""
size="90"><br>
<br>
<%
for (int t = 0 ; t < numPotentialFiles; t++)
{
%>
File: <input type="file" name="file_<%=t%>"> <br>
<%
}
%>
<br>
<hr>
User Parameters - passed through by the FTA to the
'targetPage'<br>
<hr>
Text Field:<input name="user_1" value="1">
<br>
Select Box: <select name="user_2" size="3" multiple>
<option value="one">One</option>
<option value="two">Two</option>
<option value="three">Three</option>
</select>
<input type="text" name="boris">
</form>
<hr>
<a href="#" onClick="javascript:doFileUpload();return
false;">Go</a>
</body>
</html>

```



onePartCheckinMCSForm.jsp

This form is typically in a hidden frame. (In the example, it is visible for demonstration purposes.) This form is used to send information to the ENOVIA Live Collaboration Server to get the job ticket.

```

<html>
<body>
<b>MCS Form - do not modify - this form is hidden in real
system</b>
<br>
<br>
<form name="MCSForm" action="preCheckinMCS.jsp">
NumFiles: <input name="numFiles"><br>
</form>
</body>
</html>

```

preCheckinMCS.jsp

This page calls the `preCheckin` method and returns the ticket. A JavaScript program stuffs the value of the Job Ticket and the action URL into the form in the `onePartCheckinFCSForm.jsp` page.

```
<%@ page
import="com.matrixone.servlet.* ,matrix.db.* ,com.matrixone.fcs.h
ttp.* ,com.matrixone.fcs.common.* ,com.matrixone.fcs.mcs.*"%>
<%
String processingPage = "/fcs/checkinMCS.jsp";
String targetPage = "/fcs/thankYou.jsp";
String errorPage = "/fcs/errorPage.jsp";
Context ctx = Framework.getFrameContext(session);
String store = "STORE";
int numFiles = new
Integer(request.getParameter("numFiles")).intValue();
System.out.println("Num files: " + numFiles);
TicketWrapper ticket = HttpPreCheckin.doIt(ctx,
                                             store,
                                             numFiles,
                                             processingPage,
                                             targetPage,
                                             errorPage,
                                             request,
                                             response);

String ticketStr = ticket.getExportString();
String actionURL = ticket.getActionURL();
%>
<html>
<body>
<script>
var fcsForm = parent.frames["fcsForm"].document.forms[0];
fcsForm.<%=McsBase.resolveFcsParam("jobTicket")%>.value='<%=tic
ketStr%>';
fcsForm.action='<%=actionURL%>';
fcsForm.submit();
</script>
</body>
</html>
```

checkinMCS.jsp

`checkinMCS.jsp` is called with a job receipt by the FCS when the checkin operation is completed. This page performs the actual meta data checkin.

```
<%@ page
import="matrix.db.* ,com.matrixone.fcs.http.* ,com.matrixone.fcs.
mcs.* ,com.matrixone.servlet.* ,java.util.*"
errorPage="errorPage.jsp"%>
<%@ include file="objSetup.inc"%>
<%
Context ctx = Framework.getFrameContext(session);
String store="STORE";
String format="generic";
String unlock = request.getParameter("unlock");
String append = request.getParameter("append");
String user = request.getParameter("boris");
ArrayList list = new ArrayList();
BusinessObjectProxy bop = new
BusinessObjectProxy(obj.getObjectId(),
format,
append.equals("true"),
unlock.equals("true"));

list.add(bop);
```

```
HttpCheckin.doIt(ctx,
    store,
    list,
    request,
    response);
%>
```



twoPartCheckin.jsp

This file holds the frameset used to set up the two-part checkin.

```
<%@include file="common.inc"%>
<frameset rows="40%,*" framespacing="0" frameborder="no"
border="1">
    <frame name="mcsForm" noresize marginheight="10"
marginwidth="10" border="1" scrolling="yes"
src="<%response.encodeURL("twoPartCheckinMCSForm.jsp?boid=" +
request.getParameter("boid"))%>" frameborder="0">
    <frame name="fcsForm" noresize marginheight="10"
marginwidth="10" border="1" scrolling="yes"
src="twoPartCheckinFCSForm.jsp" frameborder="0">
</frameset>
```



twoPartCheckinFCSForm

This file is the user visible form. The user uses this form to select one or more files for upload.

```
<%@include file="common.inc"%>
<%@ page
import="com.matrixone.servlet.Framework,com.matrixone.fcs.mcs.M
csBase" %>
<%
String errorPage =
Framework.getFullClientSideURL(request,response,"/fcs/
errorPage.jsp");
int numPotentialFiles = 5;
%>
<html>
<body>
<script>
function doFileUpload()
{
    var mcsForm = parent.frames["mcsForm"].document.forms[0];
    var t = 0;
<%
    for (int i = 0; i < numPotentialFiles;i++)
    {
%>
        if (document.forms["FCSForm"].file_<%=i%>.value != "")
        {
            mcsForm.fileName[<%=i%>].value =
document.forms["FCSForm"].file_<%=i%>.value;
            mcsForm.format[<%=i%>].value =
document.forms["FCSForm"].format_<%=i%>.value;
            mcsForm.append[<%=i%>].value =
document.forms["FCSForm"].append_<%=i%>[0].checked;
            mcsForm.unlock[<%=i%>].value =
document.forms["FCSForm"].unlock_<%=i%>[0].checked;
        }
<%
    }
%>
    mcsForm.submit();
}

</script>
<form enctype="multipart/form-data" method="post"
name="FCSForm" action="">
<br>
Error Page: <input
```

```

name="<%="McsBase.resolveFcsParam("failurePage")%>" value="<%="errorPage%>" size="90">><br>
JobTicket: <input name="<%="McsBase.resolveFcsParam("jobTicket")%>" value="" size="90">><br>
<br>
<%
for (int i = 0 ; i < numPotentialFiles; i++)
{
%
File: <input type="file" name="file_<%=i%>"> <br>
Format: <input name="format_<%=i%>" value="test"><br>
Append:
True: <input type="radio" name="append_<%=i%>" value="true" checked>
False: <input type="radio" name="append_<%=i%>" value="false" ><br>
Unlock:
True: <input type="radio" name="unlock_<%=i%>" value="true" checked>
False: <input type="radio" name="unlock_<%=i%>" value="false" ><br>
<%
}
%
<br>
<hr>
User Paramaters - passed through by the FCS to the 'targetPage'<br>
<hr>
Text Field:<input name="user_1" value="1">
<br>
Select Box: <select name="user_2" size="3" multiple>
<option value="one">One</option>
<option value="two">Two</option>
<option value="three">Three</option>
</select>
</form>
<hr>
<a href="#" onClick="javascript:doFileUpload();return false;">Go</a>
</body>
</html>

```



twoPartCheckinMCSForm.jsp

This form is typically in a hidden frame. (In the example, it is visible for demonstration purposes.) This form is used to send information to the ENOVIA Live Collaboration Server to get the job ticket.

```

<html>
<body>
<b>MCS Form - do not modify - this form is hidden in real system</b>
<br>
<br>
<form name="MCSForm" action="checkinMCSStart.jsp">
<%
    int numPotentialFiles = 5;
    for (int i = 0; i < numPotentialFiles;i++)
    {
%
    fileName: <input name="fileName" value="" size="50"><br>
    Boid: <input name="boid" value="<%="request.getParameter("boid")%>"><br>
    Format: <input name="format" size="50"><br>
    Append: <input name="append" size="50"><br>
    Unlock: <input name="unlock" size="50"><br>
<%
    }

```

```
%>  
</form>  
</body>  
</html>
```

checkinMCSStart.jsp

This page calls the `checkinStart` method. A JavaScript program stuffs the value of the Job Ticket and the action URL into the form in the `twoPartCheckinFCSForm.jsp` page.

```

<%@ page
import="com.matrixone.servlet.* , matrix.db.* , com.matrixone.fcs.*;
        com.matrixone.fcs.common.* , com.matrixone.fcs.mcs.* "%>
<%
void dumpList(String[] args)
{
    for (int t =0; t< args.length;t++)
    {
        System.out.println("t: " + t + "      " + args[t]);
    }
}
%>
<%
String processingPage = "/fcs/checkinMCSEnd.jsp";
String targetPage = "/fcs/thankYou.jsp";
String errorPage = "/fcs/errorPage.jsp";
Context ctx = Framework.getFrameContext(session);
String[] boids = request.getParameterValues("boid");
String[] fileNames = request.getParameterValues("fileName");
String[] formats =request.getParameterValues("format");
String[] append = request.getParameterValues("append");
String[] unlock = request.getParameterValues("unlock");
dumpList(boids);
dumpList(fileNames);
dumpList(formats);
dumpList(append);
dumpList(unlock);
String store="STORE";
TicketWrapper ticket = HttpCheckinStart.doIt(ctx,
                                              processingPage,
                                              targetPage,
                                              errorPage,
                                              store,
                                              boids,
                                              fileNames,
                                              formats,
                                              append,
                                              unlock,
                                              request,
                                              response);

String ticketStr = ticket.getExportString();
String actionURL = ticket.getActionURL();
%>
<html>
<body>
<script>
var fcsForm = parent.frames["fcsForm"].document.forms[0];
fcsForm.<%=McsBase.resolveFcsParam("jobTicket")%>.value='<=%ticketStr%>';
fcsForm.action='<%=actionURL%>';
fcsForm.submit();
</script>
</body>
</html>

```

checkinMCSEnd.jsp

This page is called by the FCS when the checkin operation is completed. This page is called with a Job Receipt. This page performs the actual meta-data checkin.

```
<%@ page
import="matrix.db.* , com.matrixone.fcs.http.* , com.matrixone.serv
let.*" errorPage="errorPage.jsp"%>
<%
Context ctx = Framework.getFrameContext(session);
String store = "STORE";
HttpCheckinEnd.doIt(ctx,
                     store,
                     request,
                     response);
%>
```



thankYou.jsp

This page is called by the ENOVIA Live Collaboration Server when the checkin is completed.

```
<html>
<body>
<script>
parent.document.location="thankYouText.jsp";
</script>
</body>
</html>
```



thankYouText.jsp

This page is called from `thankYou.jsp`.

```
<html>
<body>
Thanks for uploading with the FTA.
<br>
Back to main page
</body>
</html>
```

Checkout Page

The `checkout.jsp` page is responsible for setting up the checkout.

```
<%@ page
import="com.matrixone.servlet.* ,matrix.db.* ,com.matrixone.fcs.h
ttp.* ,com.matrixone.fcs.common.* ,com.matrixone.fcs.mcs.* "%>
<%
Context ctx = Framework.getFrameContext(session);
String errorPage = "/fcs/errorPage.jsp";
String[] boids = request.getParameterValues("boid");
String[] fileNames = request.getParameterValues("fileName");
String[] formats = request.getParameterValues("format");
String[] locks = request.getParameterValues("lock");
TicketWrapper ticket = HttpCheckout.doIt(ctx,
                                         boids,
                                         fileNames,
                                         formats,
                                         locks,
                                         errorPage,
                                         request,
                                         response);

String ticketStr = ticket.getExportString();
String ftaAction = ticket.getActionURL();
%>
<html>
<body>
<form method="post" name="FcsForm" action="<%=ftaAction%>">
<br>
<input name="<%=McsBase.resolveFcsParam( "jobTicket" )%>" value="<%=ticketStr%>" size="90"><br>
<input name="<%=McsBase.resolveFcsParam( "failurePage" )%>" value="<%=Framework.getFullClientSideURL(request,response,error
Page)%>" size="90"><br>
<input name="<%=McsBase.resolveFcsParam( "attachment" )%>" value="false" size="90"><br>
</form>
<script>
document.forms[ "FcsForm" ].submit();
</script>
</body>
</html>
```

Error Page

The `errorPage.jsp` page is the general error page. It is called by both the FCS and ENOVIA Live Collaboration Server when an error has occurred.

The `checkinMCS` and `checkinMCSStart` pages set up the error handling.

```
<%@ page isErrorPage="true" %>
<%
    String msg = request.getParameter("ErrorMessage");
    if (exception == null) {
        exception = new Exception(msg);
    }
%>
<html>
<head>
<style type="text/css" >
TABLE.error { BACKGROUND-COLOR: #f00}
TH.error {  FONT-WEIGHT: bold; FONT-SIZE: 13px; COLOR: #fff;
BACKGROUND-COLOR: #f00; TEXT-ALIGN: left}
TD.errMsg {  FONT-WEIGHT: bold; FONT-SIZE: 11px; COLOR: #f00;
BACKGROUND-COLOR: #eee}
</style>
<body bgColor=white>
<img src=images/utilSpace.gif width=1 height=15>
<TABLE class=error cellSpacing=0 cellPadding=1 width="95%" border=0 align=center>
<TBODY>
<TR>
<TD>
<TABLE cellSpacing=0 cellPadding=3 width="100%" border=0>
<TBODY>
<TR>
<TH class=error>FCS error</TH></TR>
<TR>
<TD class=errMsg><%=exception.toString() %></TD>
</TR>
</TBODY>
</TABLE>
</TD>
</TR>
</TBODY>
</TABLE>
<br>
</body>
</html>
```

FCS Network Security

This section describes the support for network security that is provided in FCS.

In this section:

- [About FCS Network Security](#)
- [About FCS Network Security Views](#)
- [Implementing Out-of-the-Box Authentication Management](#)
- [Implementing Custom Authentication Management](#)
- [FCS Network Security Error Messages](#)

About FCS Network Security

FCS provides network security support across the WAN. This includes support for Single Sign On (SSO) and network authentication.

- [SSO Scenario](#)
- [Authentication Scenario](#)

Related Topics:
[About FCS Network Security Views](#)
[Implementing Out-of-the-Box Authentication Management](#)
[Implementing Custom Authentication Management](#)
[FCS Network Security Error Messages](#)

FCS network traffic can be between a client and server, or between two servers. SSO is cookie-based and is used for signing onto a server group. Authentication, on the other hand, is header-based and is used for HTTP traffic across a specific gateway, such as a proxy. SSO can be used for client-server interaction, but it is not useful for server-server interaction. Authentication can be used for both client-server and server-server interactions. For client-server interaction, the user is expected to provide the proper credentials for either SSO or authentication. For server-server interaction, SSO is not anticipated and the credential for authentication must be provided programmatically (in other words, without user interaction).

The authentication mechanism supported in FCS is provided by the Java implementation. Currently supported authentication schemes include:

- Basic
- Digest
- NTLM
- Http SPNEGO Negotiate (defined by Microsoft)

The underlying mechanisms are Kerberos or NTLM. Credentials can be stored as an encrypted file. An authentication option is also available in the [FcsTools](#) FcsSubmit and FcsReceive.

With FCS network security support, you can:

- Configure an SSO server such that there is no longer a need for users to do multiple sign-ons for a group of servers belonging to the same entity.
- Deploy MCS and FCS servers across a WAN, where variations in proxy servers challenge the traffic.

SSO Scenario

It may be desirable to combine various systems and solutions to achieve certain goals. For example, it is inconvenient for the staff members of the same company to be required to do one sign-on per system. SSO technology allows you to deploy a unified sign-on for all of your systems and solutions. To support this scenario, [FcsClient](#) is able to operate with SSO.



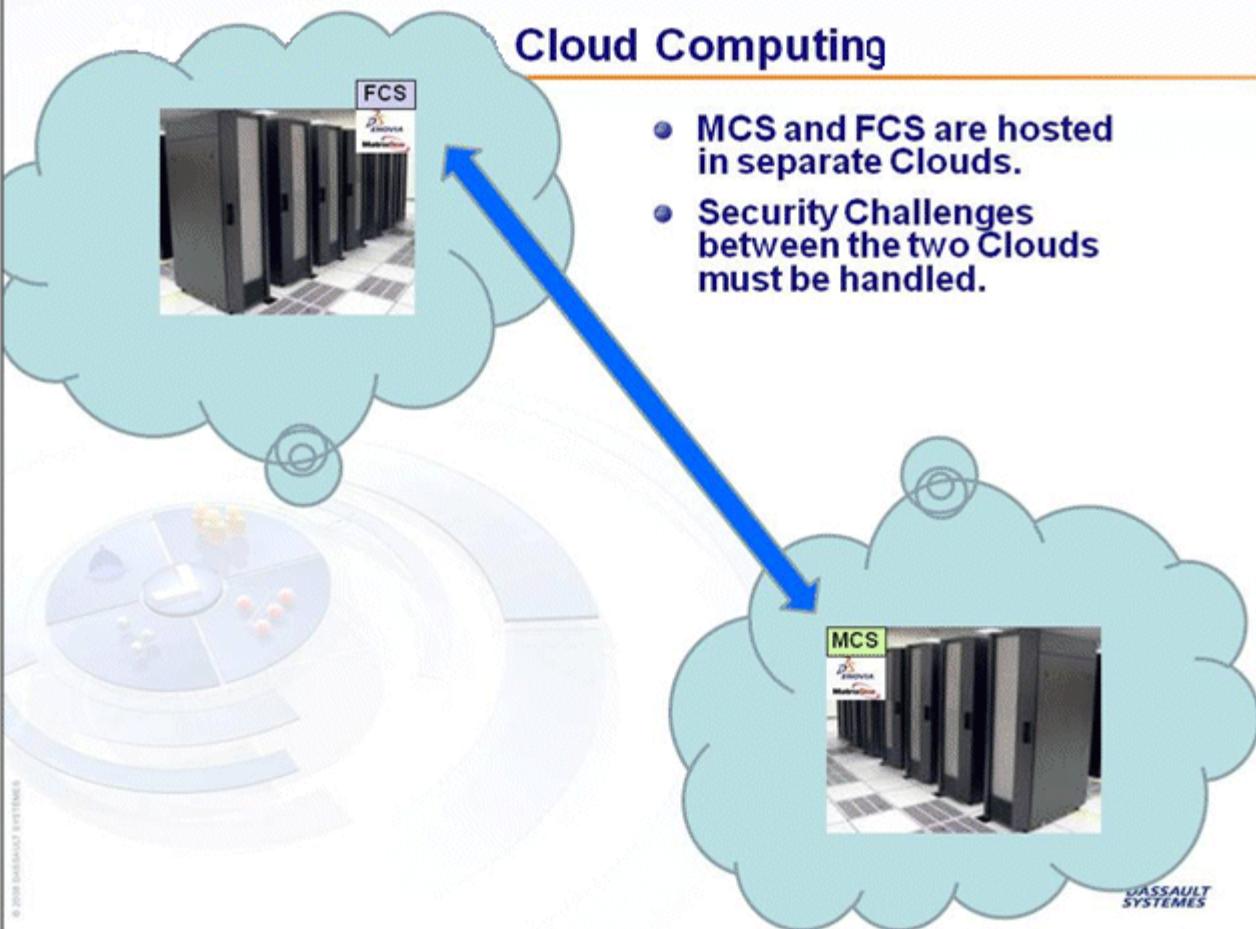
Authentication Scenario

As distributed systems become the norm, various scenarios arise in which ENOVIA must be deployed across tight security. The following are typical scenarios in which FcsClient and FCS/MCS are able to operate under the required security measures:

- Competitive partners: In a supply-chain scenario, your company may find itself working with partners who are not part of your organization or may even be competitors. In this case, network traffic between the two parties must be under tight security.
- Cloud computing: As companies move toward a total online solution, cloud computing becomes an important part of the ENOVIA platform. Physical server locations become less relevant and as a result, you may end up deploying ENOVIA servers in different clouds on different networks with different providers. In this case, network security is implied.

Cloud Computing

- MCS and FCS are hosted in separate Clouds.
- Security Challenges between the two Clouds must be handled.



About FCS Network Security Views

The user view refers to what the user sees when using their ENOVIA web application. The administration view refers to what the system administrator sees when setting up or maintaining MCS/FCS servers.

- [Client External View](#)
- [Server External View](#)
- [Client Internal View](#)
- [Server Internal View](#)

Related Topics:

- [Implementing Out-of-the-Box Authentication Management](#)
- [Implementing Custom Authentication Management](#)

Client External View

The [FcsClient](#) library does not interact directly with the user. Therefore, the user application, which uses FcsClient, is responsible for gathering the appropriate credentials from the user.



Server External View

For server-to-server interactions, authentication is the only network security method that is supported. SSO is not supported. At an MCS/FCS server, user interaction is absent and credentials must be provided programmatically. System administrators can either use the [Implementing Out-of-the-Box Authentication Management](#) or implement their own [Implementing Custom Authentication Management](#) by providing the java.net.authenticator class.



Client Internal View

The client internal view comprises SSO or authentication.

- SSO: Applications that use [FcsClient](#) for file operations are responsible for providing the cookie string containing a valid SSO cookie to FcsClient. If an SSO sign-on request is encountered (for example, 302 has temporarily moved or a cookie has already expired), FcsClient would throw an [FcsHttpResponseException](#) (see sample code below). The application should catch this exception, use `getResponseCode()` to verify the response code, and then use `getRedirection()` to obtain the URL of the SSO sign-on page. Then, the application should redirect the user to the SSO sign-on page. Once the sign-on is (re)established, the application could invoke FcsClient again with the newly acquired SSO cookie string.

```
public class FcsHTTPResponse extends Exception
{
    public int getResponseCode()
    public String getMessage()
    public String getErrorMessage()
    public String getRedirection()
    public String getMessage()
}
```

- Authentication: Applications that use [FcsClient](#) should provide a default implementation for the authenticator class. Applications that run inside a browser should automatically get the default authenticator from the browser, which can prompt the user through a dialog to enter their username and password. The application should call `Authenticator.setDefault(<authenticator_implementation>)` to set your implementation to the default authenticator. The `getPasswordCredential()` method of the authenticator class can be called whenever an authentication takes place and the credential is needed. This method should prompt the user to enter their credential, and return it accordingly. For details, see `java.net.Authenticator` and `java.net.PasswordAuthentication`.
- The [FCSTools](#) FcsSubmit and FcsRescue can be used to debug the authentication functionality. To use the default authenticator with a credentials file, use the `-authenticate` option as follows:

```
FcsSubmit ... -authenticate <credentialsfile>
```

- To use a custom authenticator, use the `-authenticator` option as follows:

```
FcsSubmit ... -authenticator <myauthenticator>
```

- Authentication: Applications that use [FcsClient](#) should provide a default implementation for the authenticator class. The application should call `Authenticator.setDefault(<authenticator_implementation>)` to set your implementation to the default authenticator. The `getPasswordCredential()` method of the authenticator class would be called whenever an authentication takes place and the credential is needed. This method should prompt the user to enter the credential, and return it accordingly. For details, see `java.net.Authenticator` and `java.net.PasswordAuthentication`.

The [FCSTools](#) FcsSubmit and FcsRescue could be used to debug the authentication functionality.

- To use the default authenticator with a credentials file, use the `-authenticate` option as follows:

```
FcsSubmit ... -authenticate <credentialsfile>
```

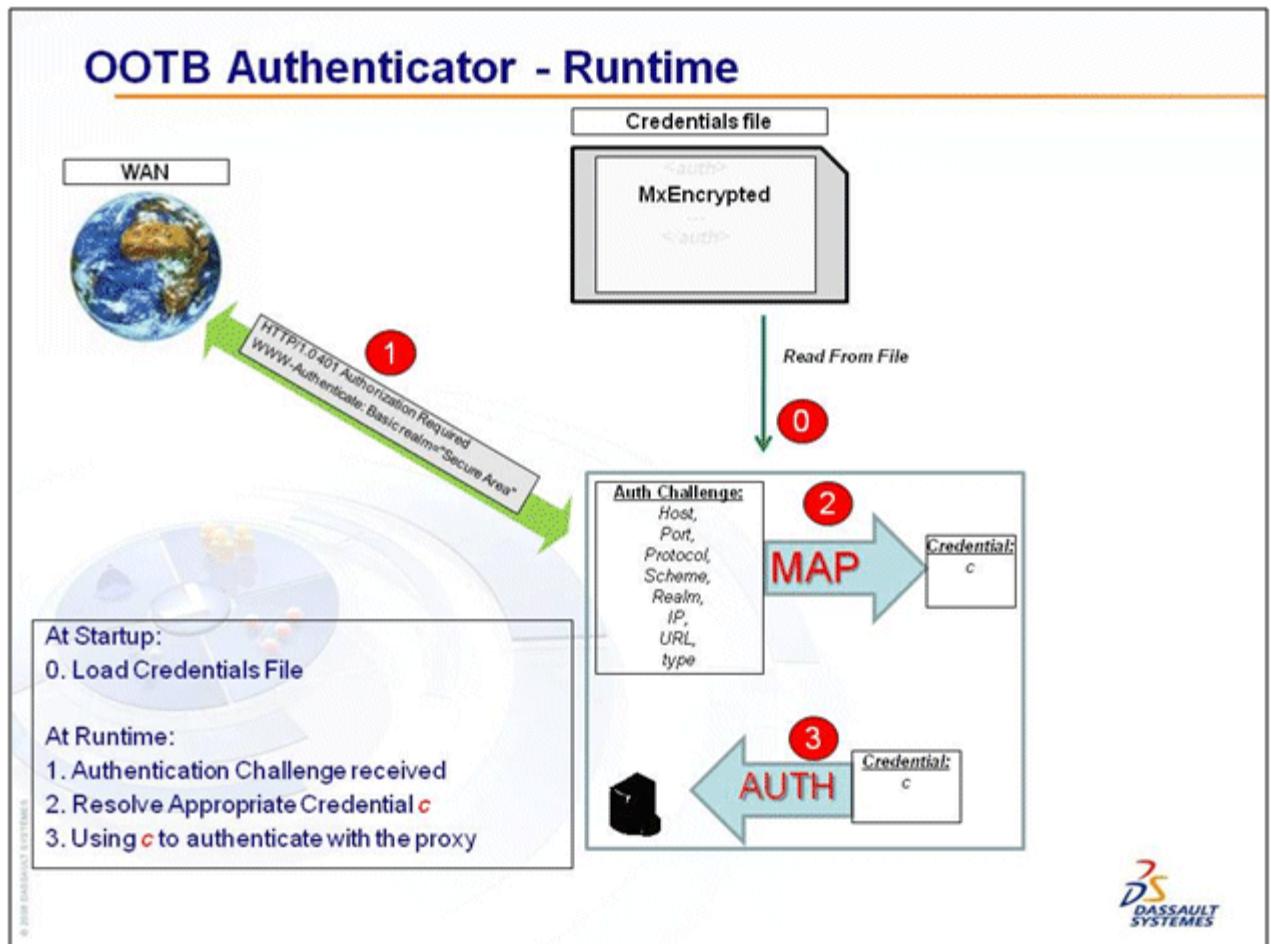
- To use a custom authenticator, use the `-authenticator` option as follows:

```
FcsSubmit ... -authenticator <myauthenticator>
```



Server Internal View

- SSO: Not supported in this view
- Authentication:
 - Custom Authentication Management: see [Server External View > Implementing Custom Authentication Management](#).
 - OOTB Authentication Management: see [Server External View > Implementing Out-of-the-Box Authentication Management](#).
 - Without Encryption: OOTB authentication management can be implemented without encryption by putting the credentials file in place and setting `MX_HTTPDEFAULTCLIENTAUTHENTICATION` to true.
 0. At startup time, the credentials file is retrieved.
 1. Upon receiving an authentication challenge, FCS first extracts the challenge information.
 2. Then, using that information, it obtains the appropriate credential.
 3. Finally, it sends the credential out for authentication.
 - With Encryption: OOTB authentication management can be implemented with encryption as follows:
 - 1. After running `MxEncryptCredentials.class` with the credentials files (and `MX_HTTPDEFAULTCLIENTAUTHENTICATION` and `MX_ENCRYPTNETWORKCREDENTIALS` both set to true), the encrypted credentials file is generated.
 0. At startup time, the credentials file is retrieved and decrypted.
 1. Upon receiving an authentication challenge, FCS first extracts the challenge information.
 2. Then, using that information, it obtains the appropriate credential.
 3. Finally, it sends the credential out for authentication.



Implementing Out-of-the-Box Authentication Management

At an MCS/FCS server, user interaction is absent and credentials must be provided programmatically. This topic describes how system administrators can implement out-of-the-box authentication management.

Before you begin: Be aware that these steps must be done with every MCS/FCS server.

Related Topics:

- [About FCS Network Security](#)
- [About FCS Network Security Views](#)
- [Implementing Custom Authentication Management](#)
- [FCS Network Security Error Messages](#)

1. Set `MX_HTTPDEFAULTCLIENTAUTHENTICATION` to true.

2. Use `MxValidateCredentials.class` to validate the format of the credentials file. The syntax is as follows:

```
java -classpath %CLASSPATH% MxValidateCredentials credentialfile
```

3. Name the credential file `mxNetworkCredentials.xml`, and place it under `%MATRIXINSTALL%/etc/`.

Note: MATRIXINSTALL is expected to be the matrix server directory (for example, `C:/enoviaav6r2011/server/`). The format is explained below.

4. To encrypt the credentials file, set `MX_ENCRYPTNETWORKCREDENTIALS` to true.

a. Use `MxEncryptCredentials.class` to create a key and encrypt the credentials file. The syntax is as follows:

```
java -classpath %CLASSPATH% MxEncryptCredentials credentialfile  
The original credentials file can now be discarded.
```

b. Optional: Use `MxEncryptCredentials.class` to reverse the credentials for verification. The syntax is as follows:

```
java -classpath %CLASSPATH% MxEncryptCredentials -reverse
```

- For encrypted credentials, it is technically OK to encrypt the credentials file once and copy it to all MCS/FCS servers, assuming that the credentials file content is the same.
- `MX_HTTPCLIENTAUTHENTICATIONCLASS` has the first precedence.
- `MX_ENCRYPTNETWORKCREDENTIALS` defaults to false.
- You must restart the server if the credentials file is modified.
- The `MxValidateCredentials.class` and `MxEncryptCredentials.class` can simply be launched inside MQL.



Example Credentials File

The credentials file is an XML-based file. The following is an example of a credentials file:

```
<auth>  
  <credential>  
    <username>creator</username>  
    <password></password>  
    <host>1.1.1</host>  
    <port>*</port>  
    <protocol>http</protocol>  
    <scheme>BASIC</scheme>  
    <prompt>80</prompt>  
    <site>*</site>  
    <url>*</url>  
    <type>proxy</type>  
  </credential>  
</auth>
```

The XML file should have a single root node called `<auth>`, and contain zero or more `<credential>` nodes.

The `<credential>` element contains exactly ten child nodes:

- `username` - the user name used to respond to the authentication challenge
- `password` - the password used to respond to the authentication challenge
- `host` - the host condition to use the username and password (* for anything)
- `port` - the port condition to use the username and password (* for anything)
- `protocol` - the protocol condition to use the username and password (http, https, or * for anything)
- `scheme` - the scheme condition to use the username and password (BASIC, DIGEST, NTLM, or * for anything)
- `prompt` - the prompt (also known as realm) condition to use the username and password (* for anything)
- `site` - the site condition to use the username and password (* for anything)
- `url` - the target URL condition to use the username and password (* for anything)

Note: The target URL is the URL to which the original request was sent.

- `type` - the type condition to use the username and password (server, proxy, or * for anything)

Note: All nodes should be declared. Use * for nodes that are not of interest. The values of these nodes should not contain non-ASCII characters.

Implementing Custom Authentication Management

At an FCS/MCS server, user interaction is absent and credentials must be provided programmatically. This topic describes how system administrators can implement custom authentication management.

Before you begin: Be aware that these steps must be done with every MCS/FCS server.

1. To use the custom authentication management option, you must implement a `java.net.authenticator` class with a custom `getPasswordAuthentication()` method.
2. Set the `MX_HTTPCLIENTAUTHENTICATIONCLASS` variable to the name of the authenticator implementation.
3. Set the authenticator class as the default authenticator.
4. Invoke the `getPasswordAuthentication()` method at the network challenge event to obtain the appropriate credential.

Note: The authenticator class should reside in the classpath. The authenticator class cannot be a JPO since it resides only in the database.

Related Topics:

[About FCS Network Security](#)
[About FCS Network Security Views](#)
[Implementing Out-of-the-Box Authentication Management](#)
[FCS Network Security Error Messages](#)

FCS Network Security Error Messages

This topic describes error messages that may occur with FCS network security.

Related Topics:

- [About FCS Network Security](#)
- [About FCS Network Security Views](#)
- [Implementing Out-of-the-Box Authentication Management](#)
- [Implementing Custom Authentication Management](#)

Error Message	Description
Custom Authenticator Not Found	The <code>MX_HTTPCLIENTAUTHENTICATIONCLASS</code> variable is set, but the corresponding Authentication class cannot be found in the JRE class scope.
Credentials File Not Found	The <code>MX_HTTPDEFAULTCLIENTAUTHENTICATION</code> variable is set to true and the <code>MX_ENCRYPTNETWORKCREDENTIALS</code> variable is set to false, but the corresponding credentials file class cannot be found in the <code>%MATRIXINSTALL%/etc/</code> directory.
Credentials File Invalid Format	The credentials file contains invalid content. Note: The credentials file should always be validated before it is loaded. When the credentials file is encrypted, <code>MxEncryptCredentials</code> should run the validation.
No Appropriate Credential	An authentication challenge has been received, but an applicable credential cannot be resolved.
Invalid Credential	An authentication challenge has been received, but the challenger rejects the resolved credential.

Coding Examples

This section shows snippets of code supplied in the framework and ENOVIA product JSPs that perform specific tasks. The code calls servlets and methods from the Studio Customization Toolkit. For more information, see the Programming Guide.

In this section:

- [!\[\]\(4c6e64d0777f710d01f2e8209364e26e_img.jpg\) Reading Property Values from the Cache](#)
- [!\[\]\(8da284c71dd8107759802a2d444fc07e_img.jpg\) Using Push and Pop to a Shadow Agent](#)
- [!\[\]\(87302bf739bb90b7314114519973925c_img.jpg\) Setting a Target Page](#)
- [!\[\]\(019b2c351b0b811d1eb6a0a7a47d7536_img.jpg\) Checking that the User is Logged In](#)
- [!\[\]\(2d8efb27238e2af332f612033ff5737b_img.jpg\) Getting the User Context](#)
- [!\[\]\(90b394566b44d1b4a0f3de0d5b354517_img.jpg\) Looking Up an Object by Symbolic Name](#)
- [!\[\]\(1935a2df351f7d1db534ad97f9defebb_img.jpg\) Opening an Object and Getting Information](#)
- [!\[\]\(b56bdbcbb9e86b1d69b4774819363a89_img.jpg\) Reading Attributes](#)
- [!\[\]\(18da89d0e2076254de4f108883432b65_img.jpg\) Checking a User's Role](#)
- [!\[\]\(0dc9a8925e3a595bb4c27849d268dac4_img.jpg\) Displaying MQL Notices](#)
- [!\[\]\(b2650241901ed04f5d99332a5faa74ec_img.jpg\) Getting Administrative Object Names from Symbolic Names](#)
- [!\[\]\(f4dad83c534553f090f1da76e8b7dfbe_img.jpg\) Getting Symbolic Names](#)

Reading Property Values from the Cache

At startup, the system loads all properties files, including custom properties, into the cache.

To read the cached value of a property, use the following bean method:

```
com.matrixone.apps.domain.util.FrameworkProperties.getProperty  
(" <property key> " );
```

where `property key` is the name of the property, such as `emxFramework.Search.UpperQueryLimit`, and the bean returns the current value of that property.

Using Push and Pop to a Shadow Agent

When a program needs to perform actions that require privileges the current user does not have, you can push the context to a shadow agent, such as the User Agent person. After performing the actions, you can then pop the context back to that of the initial user.

To invoke push/pop to User Agent in a JSP, create a Tcl program that uses the push/pop shadow agent program, called eServicecommonShadowAgent.tcl (located in ENOVIA_INSTALL/Apps/Framework/common). The following code snippets show how to call the program and then push context to the shadow agent and pop the context back after the actions are performed. Also see the Tcl script eServicecommonConnectDisconnectObjects.tcl, which lets a user connect and disconnect an object even if the user does not have connect or disconnect access for the object. This program is also located in ENOVIA_INSTALL/Apps/Framework/common.

```
#The utLoad procedure loads other tcl utilities procedures
#Start of utLoad
proc utLoad { sProgram } {
    global glUtLoadProgs env
    if { ! [ info exists glUtLoadProgs ] } {
        set glUtLoadProgs {}
    }
    if { [ lsearch $glUtLoadProgs $sProgram ] < 0 } {
        lappend glUtLoadProgs $sProgram
    } else {
        return ""
    }
    if { [ catch {
        set sDir "$env(TCL_LIBRARY)/mxTclDev"
        set pFile [ open "$sDir/$sProgram" r ]
        set sOutput [ read $pFile ]
        close $pFile
    } ] == 0 } { return $sOutput }
    set sOutput [ mql print program '$sProgram' select code dump ]
    return $sOutput
}
# end utload
#include the push/pop shadow agent file by calling the
#utLoad Procedure
eval [ utLoad eServicecommonShadowAgent.tcl]
#call the pushShadowAgent procedure to set the context to
#the shadowAgent
pushShadowAgent
#Next execute the commands the Shadow Agent needs to do
set sCmd "mql connect businessobject?.."
set sCmd2 "mql add property?.."
#If it is required that we exit on error on any command,
#popShadowAgent must be called to restore the original user
#context in the error handling code.
set mqlret [ catch {eval $sCmd} outStr]
set mqlret [ catch {eval $sCmd2} outStr]
?..
#If all commands were successful, call popShadowAgent procedure
#to restore the original user context
popShadowAgent
```

In a JSP, you invoke the program that uses eServicecommonShadowAgent.tcl via the MQL Command class. These snippets show how to invoke the eServicecommonConnectDisconnectObjects.tcl program in a JSP:

```
MQLCommand connectMQL = new MQLCommand();
    connectMQL.open(context);
    String connectMQLString = "execute program
eServicecommonConnectDisconnectObjects.tcl connect
"+sParentObjId+" "+sMarkupId+" relationship_EBOMMarkup {}";
// The tcl program eServicecommonConnectDisconnectObjects.tcl
switches the context to a shadow agent, connects the objects,
then pops the context back to the original user.
```

```
connectMQL.executeCommand(context,connectMQLString);
String result = connectMQL.getResult().trim();
String error = connectMQL.getError();
connectMQL.close(context);
```

Setting a Target Page

Use this code to set a target page.

```
String pageName = request.getRequestURI();
if (request.getQueryString() != null) {
    pageName += "?" + request.getQueryString();
}
Framework.setTargetPage(session, pageName);
```

Checking that the User is Logged In

This code checks if a user is logged in.

```
if (!Framework.isLoggedIn(request)) {  
    %>  
    <jsp:forward page ="SampleLogin.jsp"/>  
    <%  
    return;
```

Getting the User Context

This code retrieves the user context.

```
matrix.db.Context context = Framework.getContext(session);
```

Looking Up an Object by Symbolic Name

This code looks up the symbolic name for the Comments attribute.

```
<%
// load the comments attribute name
String commentsStr =
Framework.getPropertyValue(session, "attribute_Comments");

// read the comments attributes
String comments = getAttribute(session, rtsObject,
commentsStr);
%>
```

Opening an Object and Getting Information

This code is from a JSP that displays details about a business object.

```
// open the current BusinessObject
String busId = request.getParameter("busId");
BusinessObject busObj = new BusinessObject(busId);
busObj.open(context);
//get the BusinessObject TNR
String busType = busObj.getTypeName();
String busName = busObj.getName();
String busRevision = busObj.getRevision();
//get the BusinessObject basics
BusinessObjectBasics busBasics = busObj.getBasics(context);
String busOwner = busBasics.getOwner();
String busOriginated = busBasics.getCreated();
String busModified = busBasics.getModified();
String busDescription = busBasics.getDescription();
//get the current state of the businessobject
State busState = getCurrentState(session, busObj);
String busCurrentState = busState.getName();
```

Reading Attributes

This code gets the name and value for an object's attributes. The code highlighted in bold and green is the relevant code.

```
// read the related business object attributes
AttributeItr attItr = new
AttributeItr(busObj.getAttributes(context).getAttributes());
while (attItr.next()) {
    Attribute attribute = attItr.obj();
    AttributeType attrType = attribute.getAttributeType();
    attrType.open(context);
    if (colorFlag) {
        rowColor = "even";
        colorFlag = false;
    } else {
        rowColor = "odd";
        colorFlag = true;
    }
%>
<tr class="<%= rowColor%>">
<td width="50%"><%= attItr.obj().getName()%></td>
<td width="50%"><%= attItr.obj().getAttribute() %></td>
```

Checking a User's Role

This code checks to see if the current user is a buyer.

```
<%
// see if the user is a buyer
if (!isBuyer(session)) {
%>
<jsp:include page="eServiceEnd.jsp" />
<%
return;
}%>
```

Displaying MQL Notices

You can use the program `emxMQLNotice.jsp`, which is located in the `ematrix` directory, to fetch and display MQL notices.

Call the program using an include statement, like the following:

```
<jsp:include page="emxMQLNotice.jsp" flush="false"/>
```

Getting Administrative Object Names from Symbolic Names

This section describes how to use Java and Tcl to retrieve the administrative object names based on symbolic names.

The following topics are discussed:

- [Java](#)
- [Tcl](#)

Java

The method described below can be used to get administrative object names in JSPs, JavaBeans, and JPOs.

To get the name of an administrative object from the cache based on the name stored in the symbolic name administrative property, use this method:

```
PropertyUtil.getSchemaProperty("Symbolic_name");
```

For example, to get the actual administrative name for the type whose symbolic name property is type_Part, use this code:

```
String partAdminName =  
PropertyUtil.getSchemaProperty("type_Part");
```

The code would return the actual type name "Part" from the cache.

After getting the name of an administrative object, you would typically get the internationalized version of the name. For instructions, see the *Live Collaboration Administrator's Guide*.

For more information on symbolic names for administrative objects, see "How the System Identifies Administrative Objects: Symbolic Name Properties" the *Live Collaboration Administrator's Guide*.



Tcl

This section describes how to get administrative object names when writing a Tcl program, although Java is the preferred method for writing programs. You should use a JPO instead of Tcl for triggers.

The program eServiceSchemaVariableMapping.tcl contains the procedure eServiceGetCurrentSchemaName that gets the current name for an administrative object. The signature for using the procedure is:

```
proc eServiceGetCurrentSchemaName { sItemName sProgramName  
sPropertyName {sStateName ""}}
```

where:

- sItemName is the administrative object type, such as attribute, type, relationship
- sProgramProg is the name of the program on which all the administrative object's properties are registered
- sPropertyName is the administrative object's symbolic name

For example, to get the name of the administrative object whose symbolic name property is attribute-Originator:

```
set sAttrOriginator [eServiceGetCurrentSchemaName attribute  
eServiceSchemaVariableMapping.tcl attribute_Originator]
```

For example, to get a policy state name:

```
set sAttrOriginator [eServiceGetCurrentSchemaName state  
eServiceSchemaVariableMapping.tcl state_Active  
policy_eServiceTriggerProgramPolicy]
```

Getting Symbolic Names

This section shows you how to retrieve the symbolic name of an object using Java and Tcl.

The following topics are discussed:

- [Java](#)
- [Tcl](#)

Java

The method described below can be used to get symbolic names for administrative objects in JSPs, JavaBeans, and JPOs.

To get the symbolic name for an administrative object, use this method:

```
FrameworkUtil.getAliasForAdmin(context, "Type", "admin name",  
"useCache value");
```

where:

- "Type" is the administrative type, such as Type, Attribute, Relationship, etc.
- "admin name" is the name of the administrative object, such as Part or Quantity
- "useCache value" is a Boolean to get the value from the cache or from the database: True gets the value from cache and false gets it from the database

For optimum performance, obtain the symbolic name from the cache and not from the database.

Example 1

To get the symbolic name for the Part type administrative object Part from the cache, use this code:

```
String typePart = FrameworkUtil.getAliasForAdmin(context,  
"Type", "Part", true);
```

The code would return "type_Part".

Example 2

To get the symbolic name for the EBOM relationship administrative object from the cache, use this:

```
String relEBOM = FrameworkUtil.getAliasForAdmin(context,  
"Relationship", "EBOM", true);
```

The code would return "relationship_EBOM".

For more information on symbolic names for administrative objects, see "How the System Identifies Administrative Objects: Symbolic Name Properties" the *Live Collaboration Administrator's Guide*.



Tcl

This section describes how to get the symbolic name for administrative objects when writing a Tcl program, although Java is the preferred method for writing programs. You should use a JPO instead of Tcl for triggers.

The program eServiceSchemaVariableMapping.tcl contains the procedure emxGetPropertyFromAdminName that gets the symbolic name for an administrative object. The signature for using the procedure is:

```
proc emxGetPropertyFromAdminName {sAdminType sRegProg  
sAdminName}
```

where:

- sAdminType is the administrative object type, such as attribute, type, relationship
- sRegProg is the name of the program on which all the administrative object's properties are registered
- sAdminName is the current name of the administrative object whose symbolic name you want

For example, to get the symbolic name for the attribute named Originator:

```
set sAttOriginatorProperty [emxGetPropertyFromAdminName  
"attribute" eServiceSchemaVariableMapping.tcl Originator]
```

Use the emxGetPropertyFromStateName procedure to get the symbolic name for a state. the signature for using the procedure is:

```
proc emxGetPropertyFromStateName {sPolicyName sStateName}
```

where:

- sPolicyName is the name of the policy that contains the state
- sStateName is the name of the state whose symbolic name you want

For example:

```
set sStateProp [emxGetPropertyFromStateName Person Active]
```


Index

[Numerics](#) | [A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#)
| [Q](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#)

Numerics

`${COMMON_DIR}`
 `${COMPONENT_DIR}`
 `${NAME}`
 `${REVISION}`
 `${ROOT_DIR}`
 `${SUITE_DIR}`
 `${TABLE_SELECTED_COUNT}` macro
 `${TYPE}`
+ sign in tree category

A

AEFHistory command
AEFHistoryAllRevisions command
AEFHistoryAllVersions command
APIs
 emxEditableTable
API
 JavaScript
 beans
 for navigation tree
 load custom styles
 notification
Access Behavior setting
Access Expression setting
Access Function setting
Access Function setting
Access Mask setting
Access Program setting
Access parameter
Action Label setting
Action Type setting
Actions menu
Admin Type setting
Allow Manual Edit setting
Alt parameter
Alternate OID expression setting
Alternate OID
 expression setting

Alternate Policy expression setting   

Alternate Type expression setting    

AppendParameters parameter  

Application Components 

Applies To parameter  

Architecture

product 

Arguments parameter  

Arithmetic Expression setting 

Auto Filter setting   

abort method 

accessing applications externally 

access

select expression 

to form fields 

to form row 

to menu command 

to messages 

to table columns   

to tabs 

to toolbar item  

to tree categories   

to tree category 

action macros 

action trigger

described 

transaction boundaries 

addJPO parameter 

administrative objects

getting name of  

getting symbolic name of 

names of 

prefixed with 'eService' 

alternate OID

form field 

alternate tree 

alternate type icon

form field 

appendColumns parameter  

appendFields parameter 

appendURL parameter 

applet

ADK 

package source code 

applications 

applyURL parameter 

arguments

for Trigger Manager 



modifying for triggers

authentication [\[+\]](#)

auto filter [\[+\]](#) [\[+\]](#) [\[+\]](#) [\[+\]](#) [\[+\]](#) [\[+\]](#)

autoName

creating objects [\[+\]](#)

automatic compile [\[+\]](#)

average of column values [\[+\]](#) [\[+\]](#)



B

Back menu [\[+\]](#)

Building Java Applets [\[+\]](#)

Business Process Services [\[+\]](#) [\[+\]](#)

components of [\[+\]](#)

naming [\[+\]](#)

overview of [\[+\]](#)

using this guide with [\[+\]](#)

BusinessObject.getSelectBusinessObjectData method [\[+\]](#)

backslash [\[+\]](#)

beans

extending [\[+\]](#)

boolean field values [\[+\]](#)

browser

structure [\[+\]](#)

building

Java applications [\[+\]](#)

business object

DesignSync selectables [\[+\]](#)

eService Trigger Program Parameters [\[+\]](#)

identification macros [\[+\]](#)

installed with framework [\[+\]](#)

select expressions [\[+\]](#)

businessobject command [\[+\]](#)



C

C++ eMatrixMQL applications [\[+\]](#)

COMMON_DIR macro [\[+\]](#)

COMPONENT_DIR macro [\[+\]](#)

Calculate Average Label setting [\[+\]](#)

Calculate Average setting [\[+\]](#)

Calculate Custom Label setting [\[+\]](#)

Calculate Custom Program saetting [\[+\]](#)

Calculate Custom setting [\[+\]](#)

Calculate Maximum Label setting [\[+\]](#)

Calculate Maximum setting [\[+\]](#)

Calculate Median Label setting [\[+\]](#)

Calculate Median setting →
Calculate Minimum Label setting →
Calculate Minimum setting →
Calculate Standard Deviation Label setting →
Calculate Standard Deviation setting →
Calculate Sum Label setting →
Calculate Sum setting →
Calendar Function setting → → → →
Calendar Program setting → → → →
CancelButton parameter → → →
CancelLabel parameter → → →
Category setting →
Change Password command →
Channel parameter →
Client package →
Code parameter → → →
Cols setting → →
Column Count setting → →
Column Icon setting → → →
Column Type setting → → →
CommandName parameter → →
Commands parameter →
Common package →
Comparable setting → → →
Compare Report setting → → →
Confirm Message setting → → →
ContentPage parameter → →
Context.resetContext method →
ConvertCurrencyTag →
Create Exclude setting → →
Currency Converter setting → → →
Currency Expression setting → →
CurrencyConverter parameter →
CustomFilter class →
cabcelProcessJPO parameter →
cache
 getting administrative name from →
 getting symbolic name from →
calculation settings → →
calculations parameter → → →
callbackFunction parameter →
cancel processing for forms →
cancelLabel parameter → →
cancelProcessJPO parameter →
cancelProcessURL parameter → →
cascading style sheets →
channel tabs

PowerView page [\[\]](#)
channel
 controlling the layout of channels [\[\]](#)
 controlling the layout of tabs [\[\]](#)
 launch [\[\]](#)
 parameters for [\[\]](#)
chart parameter [\[\]](#) [\[\]](#)
charts in PowerView [\[\]](#)
check boxes for table columns [\[\]](#)
check box
 structure browser [\[\]](#)
check trigger [\[\]](#)
checkbox
 read only [\[\]](#)
checkout [\[\]](#)
 use applet [\[\]](#)
class names [\[\]](#)
class packages
 Client package [\[\]](#)
 Common package [\[\]](#)
 ENOVIA Live Collaboration Servlet package [\[\]](#)
 Matrix package [\[\]](#)
 Resource package [\[\]](#)
 Util package [\[\]](#)
 VUI package [\[\]](#)
 db package [\[\]](#)
 listed [\[\]](#)
classification attribute [\[\]](#) [\[\]](#)
classification path [\[\]](#) [\[\]](#)
client applications [\[\]](#)
client-side validation
 structure browser [\[\]](#)
close method [\[\]](#)
colors [\[\]](#)
columnMap [\[\]](#)
column
 calculation settings [\[\]](#) [\[\]](#)
 calculations in editable table [\[\]](#)
 count of items [\[\]](#)
 defining in editable table [\[\]](#)
 editable in structure browser [\[\]](#)
 images as headers [\[\]](#)
 units of measure [\[\]](#)
combo box [\[\]](#)
 manual entry [\[\]](#)
 sort direction [\[\]](#)
commands
 internationalizing [\[\]](#)

commit method [\[\]](#)
common components
 jar file [\[\]](#)
common directory [\[\]](#)
common schema [\[\]](#)
common.jar [\[\]](#)
compareBy parameter [\[\]](#) [\[\]](#)
compile program command [\[\]](#)
compiling Java class packages [\[\]](#)
compiling servlets [\[\]](#)
compiling [\[\]](#)
configurable form [\[\]](#)
connection macros [\[\]](#)
connectionProgram parameter [\[\]](#) [\[\]](#) [\[\]](#)
connection
 modifying [\[\]](#)
conversion preferences [\[\]](#)
count for tree category [\[\]](#)
create
 default field attributes [\[\]](#)
 form fields [\[\]](#)
 form page toolbar [\[\]](#)
 generic form [\[\]](#)
 results [\[\]](#)
css files [\[\]](#)
currency conversion [\[\]](#)
custom JSP filter [\[\]](#) [\[\]](#)
custom applications
 naming [\[\]](#)
custom filter [\[\]](#)
custom navigation tree [\[\]](#)
customize parameter [\[\]](#)



D

DSFA [\[\]](#)
Date Format setting [\[\]](#)
Decimal Precision setting [\[\]](#) [\[\]](#)
Default Category setting [\[\]](#) [\[\]](#)
Default setting [\[\]](#) [\[\]](#) [\[\]](#)
DefaultCategory parameter [\[\]](#)
Delimiter setting [\[\]](#) [\[\]](#)
Description parameter [\[\]](#) [\[\]](#)
DesignSync File Access [\[\]](#)
DesignSync
 file connection [\[\]](#)
 files [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)

- folder
- module
- server
- store
- tracing
- Diff Code setting
- Display Format setting
- Display Time setting
- Dynamic Command Function setting
- Dynamic Command Program setting
- Dynamic URL setting
- data
 - handling in Web Services
- date/time display
- dates
 - in form field
- db package
 - described
- debugging
- deprecation warnings
- default paramter
- default tree
- default values
 - create form
- defaultType
- delete relationship event
 - override trigger
- dependency rules for jars
- deprecated methods
 - for navigation trees
- deprecation warnings when compiling
- details tree
- dimension (unit of measure)
- direction parameter
- directionFilter parameter
- directory macros
- disableSorting parameter
- disabling
 - File Collaboration Server
 - event triggers
- discussion access
- displayCDMFileSummary parameter
- displayMode setting
- documentation
 - for applications
- dynamic UI components
- dynamic URL
 - in custom pages
 - in form fields

in table columns [\[\]](#)
dynamic user interface
 accessing externally [\[\]](#)
 building menus [\[\]](#)
 expressions [\[\]](#)
 macros [\[\]](#)
 naming conventions [\[\]](#)
 overview [\[\]](#)
 style sheets [\[\]](#)

E

EBOM expansion [\[\]](#)
EBOM level information [\[\]](#)
ENOVIA Live Collaboration Servlet package [\[\]](#)
ENOVIA products [\[\]](#)
 accessing externally [\[\]](#)
 documentation [\[\]](#)
 list of [\[\]](#)
Edit Access Function setting [\[\]](#)
Edit Access Mask setting [\[\]](#) [\[\]](#)
Edit Access Program setting [\[\]](#)
Edit Exclude setting [\[\]](#) [\[\]](#)
Edit link on form page [\[\]](#)
Edit mode for form [\[\]](#)
Editable setting [\[\]](#) [\[\]](#) [\[\]](#)
Effective Date Expression setting [\[\]](#) [\[\]](#)
Email Listener
 Invoking the JPO [\[\]](#)
 Troubleshooting [\[\]](#)
 functions [\[\]](#)
 mailListener.xml File [\[\]](#)
 preprocessing Email Messages [\[\]](#)
Error
 #1900068 [\[\]](#)
ExclusionList parameter [\[\]](#)
Expand Function setting [\[\]](#) [\[\]](#)
Expand Inquiry setting [\[\]](#) [\[\]](#)
Expand Program setting [\[\]](#) [\[\]](#)
Export parameter [\[\]](#) [\[\]](#) [\[\]](#)
Export setting [\[\]](#) [\[\]](#) [\[\]](#)
ExportFormat property [\[\]](#)
Expression parameter [\[\]](#) [\[\]](#)
eMatrix Java class packages [\[\]](#)
eMatrixAppletDownloadXML.jar [\[\]](#)
eMatrixMQL applications [\[\]](#)
eService Method Name attribute [\[\]](#)

eService Number Generator Object

 create form [\[+\]](#)

eService Program Argument Desc attribute [\[-\]](#)

eService Program Argument attribute [\[+\]](#)

eService Program Name attribute [\[+\]](#)

eService Sequence Number attribute [\[+\]](#)

eService Target States attribute [\[+\]](#)

eService Trigger Program Parameters

 creating object [\[+\]](#)

 for Set Originator attribute program [\[-\]](#)

 how used [\[+\]](#)

 installed with framework [\[+\]](#)

 processed by Trigger Manager [\[+\]](#)

eServiceGetCurrentSchemaName [\[+\]](#)

eServiceSchemaVariableMapping.tcl

 getting administrative object name [\[+\]](#)

 getting symbolic names [\[+\]](#)

eServiceSuiteFramework.UIForm.ValidationFile [\[+\]](#)

eServiceValidRevisionChange_if.tcl trigger program [\[-\]](#)

eServicecommonCheckRelState_if.tcl trigger program [\[-\]](#)

eServicecommonPreviousRevisionPromotion_if.tcl trigger program [\[-\]](#)

eServicecommonRequiredConnection.tcl trigger program [\[-\]](#)

eServicecommonShadowAgent.tcl [\[-\]](#)

eServicecommonTrigcRequiredFormat_if.tcl trigger program [\[-\]](#)

editLink parameter [\[+\]](#) [\[-\]](#) [\[+\]](#) [\[-\]](#)

editRelationship parameter [\[-\]](#)

editToolbar parameter [\[+\]](#) [\[-\]](#) [\[+\]](#) [\[-\]](#)

editable tables [\[-\]](#)

editor [\[-\]](#)

email

 accessing from [\[-\]](#)

emxChart.jsp

 parameters for [\[-\]](#)

emxCommonBaseComparator program [\[-\]](#)

emxCreate.jsp [\[-\]](#)

 URL parameters [\[-\]](#)

 connecting to object code [\[-\]](#)

 parameters [\[-\]](#)

 settings [\[-\]](#)

emxEditableTable APIs [\[-\]](#)

emxForm.jsp

 URL parameters [\[-\]](#)

 example URLs [\[-\]](#)

emxFormEditDisplay.jsp [\[-\]](#)

emxFramework.Preferences.ExportFormat.Choices [\[-\]](#)

emxFramework.Preferences.ExportFormat.Default [\[-\]](#)

emxFramework.Preferences.FieldSeparator.Choices [\[-\]](#)

emxFramework.Preferences.FieldSeparator.Default [\[-\]](#)

emxFramework.Preferences.FieldValueSeparator.Delimiter →
emxFramework.Preferences.Language.Choices →
emxFramework.Preferences.Language.Default →
emxFramework.Preferences.RemoveCarriageReturns.Choices →
emxFramework.Preferences.RemoveCarriageReturns.Default →
emxFramework.Preferences.ToggleBrowserToolbar default →
emxFramework.Preferences.ToggleBrowserToolbar →
emxFramework.ShowMassUpdate →
emxFramework.UIForm.ValidationFile →
emxFramework.UseDownloadApplet →
emxFramework.smallIcon.defaultType →
emxFramework.smallIcon →
emxGetPropertyFromAdminName →
emxGetPropertyFromStateName →
emxHistory.jsp →
emxIndentedTable.jsp →
 URL parameters →
emxMQLNotice.jsp →
emxNavigator.DefaultTree.TreeName →
emxNavigator.UITable.Style.Dialog → →
emxNavigator.UITable.Style.List → →
emxNavigator.jsp
 accessing externally →
 parameters for →
 parameters passed to →
emxNotificationUtil JPO →
 methods →
emxPortal.jsp →
 URL parameters →
emxSortNumericAlphaLarger program →
emxSortNumericAlphaSmaller program →
emxStructureCompare.jsp →
emxStructureCompareReport.jsp →
emxSubscriptionDialog.jsp →
emxSubscriptionPushDialog.jsp →
emxSuiteDirectory parameter → →
emxTable.jsp
 creates table page →
 parameters for →
emxTableForm →
emxTableRowId → →
emxTagLibInclude.inc →
emxTree.jsp
 parameters for →
emxTreeAlternateMenuName →
emxTriggerManager program →
emxTypeChooser.jsp →

- emxUIChooser.css [\[+\]](#)
- emxUIDialog.css [\[+\]](#)
- emxUIForm.css [\[+\]](#)
- emxUILifeCycle.css [\[+\]](#)
- emxUIList.css [\[+\]](#)
- emxUIListPF.css [\[+\]](#)
- emxUINavigator.css [\[+\]](#)
- emxUIProperties.css [\[+\]](#)
- emxUISearch.css [\[+\]](#)
- emxUIToolbar.css [\[+\]](#)
- emxUITree.css [\[+\]](#)
- emxUIWizard.css [\[+\]](#)
- emxUIWorkflow.css [\[+\]](#)
- emxcommonSetOriginator_if trigger program [\[+\]](#)
- enableSearchButton parameter [\[+\]](#) [\[-\]](#)
- enabling
 - event triggers [\[+\]](#)
- enovia.ini file
- Studio Modeling Platform [\[+\]](#)
- enovia.ini
- BOS_EXPAND_LIMIT [\[+\]](#)
- error handling for JSPs [\[+\]](#)
- error
 - message [\[+\]](#)
- event triggers
 - action program [\[+\]](#)
 - action [\[+\]](#) [\[-\]](#)
 - and recursion [\[+\]](#)
 - attribute event macros [\[+\]](#)
 - auto activity events [\[+\]](#)
 - blocking events [\[+\]](#)
 - business object event macros [\[+\]](#)
 - check [\[+\]](#)
 - create event [\[+\]](#)
 - disabling [\[+\]](#)
 - enabling [\[+\]](#)
 - interactive activity events [\[+\]](#)
 - lifecycle actions [\[+\]](#)
 - lifecycle checks [\[+\]](#)
 - multi-trigger events [\[+\]](#)
 - override on disconnect [\[+\]](#)
 - override [\[+\]](#) [\[-\]](#)
 - process event macros [\[+\]](#)
 - process events [\[+\]](#)
 - programs [\[+\]](#)
 - recursion [\[+\]](#)
 - relationship event macros [\[+\]](#)
 - relationship [\[+\]](#)

- replacing events [\[+\]](#)
- scenarios [\[+\]](#)
- state event macros [\[+\]](#)
- state events [\[+\]](#)
- transaction boundaries [\[+\]](#)
- transaction model [\[+\]](#)
- types [\[+\]](#)
- user interaction in [\[+\]](#)
- workflow events [\[+\]](#)
- events
 - logging [\[+\]](#)
- example
 - hello world [\[+\]](#)
- excludeOID parameter [\[+\]](#)
- excludeOIDprogram parameter [\[+\]](#) [\[+\]](#)
- execution sequence
 - modifying [\[+\]](#)
- expandLevelFilter parameter [\[+\]](#) [\[+\]](#)
- expandLevelFilterMenu parameter [\[+\]](#) [\[+\]](#)
- expandProgram parameter [\[+\]](#) [\[+\]](#) [\[+\]](#)
- expandProgramMenu parameter [\[+\]](#) [\[+\]](#)
- expand
 - streaming [\[+\]](#)
- export format [\[+\]](#)
- expressions used in dynamic UI [\[+\]](#)
- extending
 - JSPs [\[+\]](#)
 - beans [\[+\]](#)
- external access [\[+\]](#)
- extract program command [\[+\]](#) [\[+\]](#)
- extracting Java source [\[+\]](#)



F

- FCS:and large file handling [\[+\]](#)
- FCS:zip support [\[+\]](#)
- FTPS [\[+\]](#)
- FTP [\[+\]](#)
- Field Column Headers setting [\[+\]](#) [\[+\]](#)
- Field Row Headers setting [\[+\]](#) [\[+\]](#)
- Field Size setting [\[+\]](#) [\[+\]](#) [\[+\]](#)
- Field Table Columns setting [\[+\]](#) [\[+\]](#)
- Field Table Rows setting [\[+\]](#) [\[+\]](#)
- Field Type setting [\[+\]](#)
 - for form object fields [\[+\]](#)
 - for table object columns [\[+\]](#) [\[+\]](#)
- Filter tool [\[+\]](#)

- FilterFramePage parameter [\[\]](#) [\[\]](#)
- FilterFrameSize parameter [\[\]](#) [\[\]](#)
- Format parameter [\[\]](#) [\[\]](#)
- FormatURL taglib [\[\]](#)
- Forward menu [\[\]](#)
- Framework Components [\[\]](#)
- field parameter [\[\]](#) [\[\]](#)
- fieldLabels parameter [\[\]](#)
- fieldNameActual parameter [\[\]](#) [\[\]](#)
- fieldNameDisplay parameter [\[\]](#) [\[\]](#)
- fieldSeparator parameter [\[\]](#) [\[\]](#)
- fields
 - arithmetic expressions [\[\]](#)
- file
 - install log file [\[\]](#)
- filter toolbar
 - structure browser [\[\]](#)
- filterColumnPosition parameter [\[\]](#) [\[\]](#)
- filtering JSPs [\[\]](#) [\[\]](#)
- filter
 - custom for table [\[\]](#)
 - tool [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
- findMxLink parameter [\[\]](#) [\[\]](#) [\[\]](#)
- findMxLink setting [\[\]](#)
- finding ENOVIA object ID from DesignSync [\[\]](#)
- fonts [\[\]](#)
- form page [\[\]](#)
 - Edit link [\[\]](#)
 - Edit mode [\[\]](#)
 - URL for [\[\]](#)
 - View mode [\[\]](#)
 - committing changes [\[\]](#)
 - creating [\[\]](#)
 - decision and options for fields [\[\]](#)
 - edit and refresh [\[\]](#)
 - html components [\[\]](#)
 - linking to application [\[\]](#)
 - links [\[\]](#)
 - mechanisms that define [\[\]](#)
 - refreshing after edit [\[\]](#)
 - two modes [\[\]](#)
- form parameter [\[\]](#) [\[\]](#) [\[\]](#)
- formHeader parameter [\[\]](#) [\[\]](#)
- formInclusionList parameter [\[\]](#) [\[\]](#)
- formName parameter [\[\]](#)
- formViewDisplay frame [\[\]](#)
- formViewHeader frame [\[\]](#)

formViewHidden frame [\[\]](#)
format macros [\[\]](#)
format setting [\[\]](#) [\[\]](#)
form [\[\]](#) [\[\]](#)
 JPO for field values [\[\]](#)
 cancel processing [\[\]](#)
 committing changes [\[\]](#)
 date/time display [\[\]](#)
 decision and options for fields [\[\]](#)
 field [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
 generic create [\[\]](#)
 manual entry for combo box [\[\]](#)
 parameters for [\[\]](#)
 post processing [\[\]](#)
 pre processing [\[\]](#)
 using Action menus with [\[\]](#)
frameName parameter [\[\]](#) [\[\]](#)
frameName [\[\]](#)
framework.jar [\[\]](#)
framework [\[\]](#)
freeze pane column [\[\]](#)
freezePane parameter [\[\]](#) [\[\]](#)
frmFormView form [\[\]](#)
fully qualified class references [\[\]](#)
function for table column [\[\]](#)
function setting [\[\]](#) [\[\]](#) [\[\]](#)

↑

G

Group Count setting [\[\]](#)
Group Header setting [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
Group Name setting [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
genericDelete parameter [\[\]](#) [\[\]](#)
getAliasForAdmin [\[\]](#)
getSchemaProperty [\[\]](#)
getSelectBusinessObjectData method [\[\]](#)
global toolbar
 overview [\[\]](#)
grouped fields [\[\]](#)
grouping table columns [\[\]](#)

↑

H

HTML
 source for JSP [\[\]](#)
Header parameter
 [\[\]](#)

for emxHistory.jsp
Heading parameter
Height parameter
Help Marker setting
HelpMarker parameter
Hide Label setting
History commands
History page
HistoryMode parameter
Href parameter
header parameter
for emxPortal.jsp
headerRepeat parameter
hidden fields
 create form
hiddenFrame frame
hideHeader parameter
hideRootSelection parameter
hideToolbar parameter
home page preference
href parameter
html components
 of form page
html form
hyperlink for form field
hyperlinked image

IDE
Icon parameter
IconMailLanguagePreference property
Image Size setting
Image setting
InclusionList parameter
Input Control Direction setting
Input Type setting
icons
 column values
 overriding default
icon
 column type
 for table columns
 for toolbar tools
 type's tree menu
images
 for table columns

- in column heading [\[\]](#)
- image
 - configuring images [\[\]](#)
 - for toolbar tools [\[\]](#)
- in-cell editing
 - structure browser [\[\]](#)
- includeOID program parameter [\[\]](#)
- inheritance
 - subscriptions [\[\]](#)
- inquiry parameter [\[\]](#) [\[\]](#)
- inquiry setting [\[\]](#)
- inquiryLabel parameter [\[\]](#) [\[\]](#)
- inquiry
 - parameters for [\[\]](#)
 - used to build table [\[\]](#)
 - using for expanding tree category [\[\]](#)
- insert program command [\[\]](#) [\[\]](#)
- install log files
 - lists name collisions [\[\]](#)
- installFramework.log
 - lists name collisions [\[\]](#)
- internal object [\[\]](#)
- internationalizing
 - commands [\[\]](#)
 - menus [\[\]](#)
- intrinsic macros [\[\]](#)
- isTransactionActive method [\[\]](#)



J

- JPO macros [\[\]](#)
- JPO
 - access through Studio Customization Toolkit [\[\]](#)
 - and trigger manager [\[\]](#)
 - for form field values [\[\]](#)
 - for getting field range values [\[\]](#)
 - for getting field values [\[\]](#)
 - for getting list of objects [\[\]](#)
 - for structure tree [\[\]](#)
 - for table column [\[\]](#) [\[\]](#)
 - function for table column [\[\]](#)
 - getting names of administration objects [\[\]](#)
 - getting names of administrative objects [\[\]](#)
 - getting symbolic names of administrative objects [\[\]](#)
 - guidelines for writing [\[\]](#)
 - interface for form fields [\[\]](#)
 - post processing for forms [\[\]](#) [\[\]](#) [\[\]](#)
 - program for table column [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)

sample for controlling access

sample for form field values

to update column values

to update table values

JSP programs

error handling

getting names of administration objects

getting names of administrative objects

getting symbolic names of administrative objects

sample for custom table filter

JSP

HTML source

accessing

filtering custom

multi-threaded

processing

Java Code

organization of

Java compiler

Java source

extracting

JavaBeans

JavaScript validation

custom

standard

Javadocs

documented packages

Javascript Include setting

Java

applications

class package

jar files

jar file

application

breakup

common

dependencies

framework.jar

installed with apps

installed with framework

jpoAppServerParamList parameter

jsTreeID parameter



L

LASTREVISION parameter

LATESTREVISION parameter

Label Function setting

Label Program setting
Label parameter
Label setting
Level setting
Logout command
language
 preferred for notifications
large files
launched parameter
launch
level parameter
lifecycle check macros
listHidden frame
log files
 lists name collisions
lookupJPO parameter

M

MQL command
 compile program
 extract program
 insert program
MQL notice
MQLCommand
MX_DS_TRACE
MX_EDITOR_PATH
MX_EDITOR
MX_JAVAC_FLAGS
MX_JAVAC_OPTIONS
MX_JAVA_DEBUG
MX_PASSWORD_TRIGGER
MX_SITE_PREFERENCE
MX_TRIGGER_RECURRENCE_DETECTION setting
MX_TRIGGER_RECURRENCE_DETECTION
MX_TRIGGER_RECURRENCE_LIMIT setting
Mass Update setting
Matrix Navigator
 configuring triggers
Matrix package
 described
 source code
Maximum Length setting
 for form object field
 for toolbar menu object
MenuName parameter
Menus parameter

Message URL Label setting  

Mode setting  

Mouse Over Popup setting 

My Desk menu (DS button) 

macro processing 

macros

JPO 

action 

attribute events 

business object events 

business object identification 

connection 

format 

lifecycle check 

process events 

program range 

relationship events 

state events 

workflow 

macro

for common directory 

for directory names 

for number of selected rows in table 

for object name 

for object revision 

for object type 

for root directory 

for select expressions 

for suite directory 

mandatorySearchParam parameter  

markup

loading 

rows 

mass update toolbar

structure browser 

massPromoteDemote parameter   

massUpdate parameter   

matchBasedOn parameter  

maximum column value  

median column value  

menu mode 

menus

building in dynamic UI 

internationalizing 

right-click 

menu 

Back 

method

- for refreshing table [\[+\]](#)
- minRequiredChars parameter [\[+\]](#)
- minimum column value [\[+\]](#) [\[-\]](#)
- modality [\[+\]](#) [\[-\]](#) [\[+\]](#) [\[-\]](#)
- mode parameter [\[+\]](#) [\[-\]](#) [\[+\]](#) [\[-\]](#)
 - for emxForm.jsp [\[+\]](#)
 - for emxNavigator.jsp [\[+\]](#) [\[-\]](#)
 - for emxTree.jsp [\[+\]](#)
- multi-trigger event
 - create events [\[+\]](#)
 - relationship events [\[+\]](#)
- multiColumnSort parameter [\[+\]](#) [\[-\]](#) [\[+\]](#)
- mxLink
 - in custom pages [\[+\]](#)
 - in form fields [\[+\]](#)
 - in table columns [\[+\]](#)
 - validation [\[+\]](#)



N

- NAME macro [\[+\]](#)
- Name Field setting [\[+\]](#)
- Name parameter [\[+\]](#) [\[-\]](#) [\[+\]](#)
- Navigator page
 - accessing externally [\[+\]](#)
 - configuring [\[+\]](#)
 - defining default home page [\[+\]](#)
- Nowrap setting [\[+\]](#) [\[-\]](#)
- name field
 - create form [\[+\]](#)
- name macro [\[+\]](#)
- name mangling [\[+\]](#) [\[-\]](#)
- nameField parameter for emxCreate.jsp [\[+\]](#)
- names
 - administrative objects [\[+\]](#) [\[-\]](#)
 - conventions [\[+\]](#)
 - custom applications [\[+\]](#)
 - of applications programs [\[+\]](#)
 - symbolic name of admin objects [\[+\]](#)
 - trigger programs [\[+\]](#)
- navigation trees
 - + sign [\[+\]](#)
 - API for [\[+\]](#)
 - building [\[+\]](#) [\[-\]](#)
 - calling alternate for a type [\[+\]](#)
 - calling custom [\[+\]](#)
 - configuring dynamic expand for category [\[+\]](#)
 - configuring structure tree [\[+\]](#)

count for category [\[\]](#)
default tree [\[\]](#)
deprecated methods [\[\]](#)
overview [\[\]](#)
parameters for categories [\[\]](#) [\[\]](#)
parameters for [\[\]](#)
settings for categories [\[\]](#) [\[\]](#)
sub-tree [\[\]](#)
nested transactions [\[\]](#)
newing up classes [\[\]](#)
nfs [\[\]](#)
notice
 MQL [\[\]](#)
notifications
 subscribing to [\[\]](#)
notification
 object attributes for subscription [\[\]](#)
 sample JPO for attributes [\[\]](#)
 sample XML body HTML [\[\]](#)
 samply XML body text [\[\]](#)



O

Object List Map [\[\]](#)
Object PowerView
 linking page to ENOVIA products [\[\]](#)
ObserveHidden parameter [\[\]](#) [\[\]](#)
OnChange Handler setting [\[\]](#) [\[\]](#)
OnFocus Handler setting [\[\]](#) [\[\]](#)
Originator attribute
 program that sets [\[\]](#)
Overview
 JSP [\[\]](#)
object tree. See navigation trees. [\[\]](#)
object type
 PowerView page [\[\]](#)
objectBased parameter [\[\]](#) [\[\]](#)
objectId parameter [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
objects [\[\]](#)
onReset [\[\]](#) [\[\]](#)
open method [\[\]](#)
override trigger [\[\]](#)
owner parameter [\[\]](#)
owner
 create form [\[\]](#)



P

Pattern parameter
Popup Modal setting
PowerView page
 channel tabs
 for object
 mechanisms that define
 overview
PowerView
 building
 launch
 linking page to VCP
PreExpand Category setting
PreExpand Function setting
PreExpand setting
Preferences.FieldValueSeparator.Delimiter
Printer Friendly setting
PrinterFriendly parameter
Program setting
Pull Right setting
packages
 Resource
 Util
 VUI
 db
 listed
 with source code
page toolbar
 structure browser
pageSize parameter
pagination controls
pagination parameter
parameters automatically passed to URLs
parameters for
 channel objects
 emxChart.jsp
 emxIndentedTable.jsp
 emxNavigator.jsp
 emxTable.jsp
 emxTree.jsp
 forms
 inquiry objects
 navigation trees
 portal menu objects
 structure browser
 tables
 tabs
 tree categories

parentOID parameter

performance of table columns 

policy parameter 

policy

 create form 

popup range helper

 field with 

 implementing 

popup windows    

portal mode 

portal object

 parameters for 

portal parameter   

portalMode parameter    

portabletable parameter  

post processing

 forms 

postProcessJPO parameter    

postProcessURL parameter     

pre processing for forms 

pre-processing for tree category 

preFilter parameter 

preProcessJPO parameter     

preProcessJavaScript parameter  

preProcessURL parameter   

preference_Command property 

preference_FieldSeparator property 

preference_Menu property 

preference_RecordSeparator property 

preference_RemoveCarriageReturns property 

preferences:site 

preference

 currency conversion 

 export table data format 

 home page 

 language 

 unit of measure 

private 

program parameter    

program range macros 

program setting  

programHTMLOutput

 code sample for emxCreate.jsp 

 create form 

programLabel parameter  

programs

 do not change names 

 emxTriggerManager 

for table columns [\[\]](#)
prefixes for names [\[\]](#)
trigger [\[\]](#)
program
 action trigger [\[\]](#)
 check trigger [\[\]](#)
 command syntax [\[\]](#)
 macro processing in [\[\]](#)
 override trigger [\[\]](#)
 use of backslash [\[\]](#)
properties files [\[\]](#)
public [\[\]](#)
push and pop to Shadow Agent [\[\]](#)



Q

queryLimit parameter [\[\]](#) [\[\]](#)
queryType parameter [\[\]](#) [\[\]](#)
query
 streaming [\[\]](#)



R

RMB Menu setting [\[\]](#) [\[\]](#) [\[\]](#)
RMI
 talking directly to [\[\]](#)
ROOT_DIR [\[\]](#)
RPE [\[\]](#)
Range Display Values setting [\[\]](#) [\[\]](#)
Range Function setting [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
Range Program setting [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
 for form object fields [\[\]](#)
Range Values setting [\[\]](#) [\[\]](#)
RangeHref parameter [\[\]](#)
Read Only Checkbox setting [\[\]](#) [\[\]](#)
Registered Suite setting [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
RegisteredDirectory parameter [\[\]](#) [\[\]](#)
Relationship Filter setting [\[\]](#)
Reload Function setting [\[\]](#) [\[\]](#)
Reload Program setting [\[\]](#) [\[\]](#)
ReloadOpener parameter [\[\]](#)
Remove Range Blank setting [\[\]](#)
Required setting [\[\]](#) [\[\]](#) [\[\]](#)
Resource package [\[\]](#)
 source code [\[\]](#)
Revision Filter setting [\[\]](#)
 [\[\]](#) [\[\]](#)

Root Label parameter
Row Number parameter 
Row Select setting  
Row Span setting 
Rows setting  
radio buttons 
recursion 
refreshTablePage method 
relID parameter 
relId parameter   
relationship parameter   
relationshipFilter parameter  
relationship 
 direction in select expressions 
 select expression 
 triggers
rememberSelection parameter  
renderPDF parameter   
reportType parameter  
resequenceRelationship parameter  
reset context limitation 
resultsToolbar parameter 
resultsToolbar
revision macro
right-click menus
root directory macro 
row
 expand
runtime program environment 



S

SUITE_DIR 
Section Level setting  
SelectAbstractTypes parameter 
SelectType parameter 
Selectable in Preferences setting   
Settings parameter   
Shadow Agent group
 push and pop to 
Show Alternate Icon setting   
Show Clear Button setting  
Show Type Icon setting   
ShowFilterAction parameter 
ShowFilterTextBox parameter 
ShowIcons parameter 
Sort Direction setting  

- Sort Program setting [\[\]](#) [\[\]](#)
- Sort Range Values setting [\[\]](#) [\[\]](#) [\[\]](#)
- Sort Type setting [\[\]](#) [\[\]](#) [\[\]](#)
- Sortable setting [\[\]](#) [\[\]](#)
- Sourcing Central
 - currency conversion [\[\]](#)
- String Properties [\[\]](#)
- StringResourceFileId parameter [\[\]](#) [\[\]](#)
- StringResourceFileId [\[\]](#)
- Structure Browser
 - pre/post/cancel processing [\[\]](#)
- Structure Function setting [\[\]](#)
- Structure Inquiry setting [\[\]](#)
- Structure Menu setting [\[\]](#) [\[\]](#)
- Structure Navigator [\[\]](#) [\[\]](#)
- Structure Program setting [\[\]](#)
- Studio Customization Toolkit
 - MQLCommand [\[\]](#)
 - access to JPOs through [\[\]](#)
 - overview [\[\]](#)
- Style Column setting [\[\]](#) [\[\]](#)
- Style Function setting [\[\]](#) [\[\]](#)
- Style Program setting [\[\]](#) [\[\]](#)
- Style parameter [\[\]](#) [\[\]](#) [\[\]](#)
- Submit Function setting [\[\]](#)
- Submit Program setting [\[\]](#)
- Submit setting [\[\]](#) [\[\]](#)
- SubmitLabel parameter [\[\]](#) [\[\]](#) [\[\]](#)
- SubmitURL parameter [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
- SuiteKey parameter [\[\]](#)
- System Error
 - #1500573 [\[\]](#)
- searchCollectionEnabled parameter [\[\]](#) [\[\]](#)
 - search
 - configurable webform [\[\]](#)
 - section header [\[\]](#)
 - section separator [\[\]](#)
 - select expression macros [\[\]](#)
 - select expressions
 - for dynamic UI [\[\]](#)
 - for form fields [\[\]](#)
 - for table columns [\[\]](#)
 - select expression
 - defined [\[\]](#)
 - definition objects [\[\]](#)
 - examples [\[\]](#)
 - notation [\[\]](#)
 - relationship direction [\[\]](#)

syntax 

selectable expressions

 attributes 

 business objects 

 format 

 forms 

 group 

 location 

 person 

 policy 

 process 

 program 

 query 

 role 

 store 

 table columns 

 tables 

 type 

 user 

 vault 

 wizard 

selectable fields

 for workflows 

selection parameter     

separator

 for table columns 

server-side code 

servlet Studio Customization Toolkit

 packages in 

 using to build client applications 

 using to build server-side applications 

servlets

 accessing 

 compiling 

 processing 

settings for

 structure browser 

 tables  

 tabs 

 tree categories  

settings 

shell command

 and macro processing 

shortcut menus 

showApply parameter  

showClipboard parameter   

showInitialResults parameter 

showPageHeader parameter  

showPageURLIcon parameter → [] [] [] [] [] [] []
showRMB parameter → [] [] []
showSavedQuery parameter → [] []
showTabHeader parameter → [] [] []
site:setting a site preference → []
smallIcon → []
sortColumnName parameter → [] [] [] []
sortColumnName setting → [] []
sortDirection parameter → [] [] []
sortDirection setting → []
sorting a combo box → []
sorttype parameter → []
source HTML → []
source code → []
 Matrix package → []
 Resource package → []
 applet package → []
source to file → []
standard deviation column value → [] []
start method → []
stateless objects → []
stopOOTBRefresh parameter → []
store → []
streaming
 query and expand → []
structure browser → []
 JS APIs for validation → []
 cell child IDs → []
 cell details → [] []
 cell edit → []
 cell parent ID → []
 cell value → [] [] [] []
 check box disabling → []
 client-side validation → []
 columns → []
 edit mode → []
 editable columns → []
 expand parameters → []
 expanding objects → []
 filter toolbar → []
 filters → []
 freeze pane column → []
 header display → []
 in-cell editing → []
 mass update toolbar → []
 page toolbar → []
 → []

parent column values

row [\[\]](#) [\[\]](#)

settings [\[\]](#)

table display [\[\]](#)

structure tree

API for [\[\]](#)

JPO program [\[\]](#)

configuring [\[\]](#)

described [\[\]](#)

shown [\[\]](#)

style sheets [\[\]](#)

style sheet

for table pages [\[\]](#) [\[\]](#)

styles

custom [\[\]](#)

sub-menu [\[\]](#)

sub-tree [\[\]](#)

subHeader parameter for portal [\[\]](#)

subHeader parameter for table [\[\]](#) [\[\]](#)

subHeader parameter

for emxHistory.jsp [\[\]](#)

for emxTable.jsp [\[\]](#)

submitAction parameter [\[\]](#) [\[\]](#)

submitLabel parameter [\[\]](#) [\[\]](#)

subscriptions [\[\]](#)

inheritance [\[\]](#)

notifications [\[\]](#)

options dialog [\[\]](#)

pushed [\[\]](#)

to a single object [\[\]](#)

to multiple objects [\[\]](#)

suite directory macro [\[\]](#)

suiteKey parameter [\[\]](#)

sum of column values [\[\]](#) [\[\]](#)

symbolic name

getting administrative object name from [\[\]](#) [\[\]](#)

getting for admin object [\[\]](#)

sync businessobjectlist [\[\]](#)

synchronization

explicit [\[\]](#)

implicit or on-demand [\[\]](#)

synchronizing ENOVIA updates with DesignSync [\[\]](#)

T

Target Location setting [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)

Tcl programs

getting names of administrative objects [\[\]](#)

getting symbolic names of administrative objects →
Test parameter →
Tip Page setting → → → →
TipPage parameter → → → → →
Toolbar menu →
TransactionType parameter → → →
Tree Scope ID setting → →
Trigger Manager →
Type Chooser →
Type Icon Function setting → → → → →
Type Icon Program setting → → → → →
Type parameter →
TypeAhead Character Count setting → →
TypeAhead Function setting → →
TypeAhead Program setting → →
TypeAhead Saved Values Limit setting →
TypeAhead setting → →
table page →
 building →
 how linked to application →
 mechanisms that define →
 overview →
 pagination →
 specific to business object →
 style sheet → →
table parameter → → → → → →
table selected count macro →
table setting → →
tableMenu parameter → →
table →
 JPO for column →
 JPO for getting objects →
 body definition →
 check boxes for columns →
 column calculation settings → →
 column calculations →
 configuring editable →
 count of items in column →
 custom filter →
 date/time display →
 editable → → → →
 embedding in form →
 filtering → → → → →
 getting object Id from →
 grouping columns →
 icons for columns →

- images for columns [\[\]](#)
- images in column headers [\[\]](#)
- improving performance of [\[\]](#)
- parameters for [\[\]](#) [\[\]](#)
- programs for column values [\[\]](#)
- refreshing to update [\[\]](#)
- select expressions [\[\]](#)
- selected count macro [\[\]](#)
- settings for [\[\]](#) [\[\]](#)
- special information [\[\]](#)
- updating with JPO [\[\]](#)

tabs

- controlling the layout in channel [\[\]](#)
- parameters for [\[\]](#) [\[\]](#)
- settings for [\[\]](#)

tag library [\[\]](#)

taglib

- units of measure [\[\]](#)

talking directly to RMI [\[\]](#)

toolbar parameter [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#) [\[\]](#)
toolbar

- Change Password tool [\[\]](#)

- Logout tool [\[\]](#)

- example input control commands [\[\]](#) [\[\]](#) [\[\]](#)

- in editable table [\[\]](#)

tracing

- ds [\[\]](#)

transaction boundaries

- event triggers [\[\]](#)

transactions

- nested [\[\]](#)

translating

- commands [\[\]](#)

- menus [\[\]](#)

tree categories. See navigation trees. [\[\]](#)

tree categories [\[\]](#) [\[\]](#) [\[\]](#)

tree mode [\[\]](#)

treeLabel parameter [\[\]](#) [\[\]](#)

treeMenu parameter [\[\]](#) [\[\]](#) [\[\]](#)

tree

- dynamic categories [\[\]](#)

trigger event

- adding trigger for [\[\]](#)

trigger off [\[\]](#)

trigger on [\[\]](#)

trigger programs

- Check Relative State [\[\]](#)

- Set Originator attribute [\[\]](#)

- do not change code [\[\]](#)

- do not change file names
- installed with applications
- modifying argument values for
- previous revision promotion
- required connection check
- required file check
- valid revision change
- triggerValidation parameter
- trigger
 - adding
 - modifying
 - notification API
- troubleshooting
 - DesignSync File Access
- txtExcludeOIs parameter
- txtTextSearch parameter
- type ahead
- type field
 - create form
- type icon for form field
- type macro
- type parameter
- typeChooser parameter
- typeFilter parameter



U

- UI level 3
- UNC paths
- UOM Expression setting
- UOMAttribute class
- URL parameters
 - emxIndentedTable.jsp
- URL
 - post processing for forms
 - post processing for tables
- Update Function setting
- Update Program Arguments setting
- Update Program setting
- Update URL parameter
- UseBean tag
- Using FCSTools
- Util package
- unit of measure conversion
- units of measure
 - field values
 - taglib

V

VUI package [\[\]](#)
Validate Type setting [\[\]](#) [\[\]](#) [\[\]](#)
Validate setting [\[\]](#) [\[\]](#) [\[\]](#)
Vertical Group Name setting [\[\]](#) [\[\]](#)
Vertical Group Name [\[\]](#)
View Exclude setting [\[\]](#) [\[\]](#)
View mode for form [\[\]](#)
validating form field data [\[\]](#)
 using custom JavaScript [\[\]](#)
 using standard JavaScript validation [\[\]](#)
vault parameter [\[\]](#)
vaultChooser parameter [\[\]](#)
vault
 create form [\[\]](#)
vcfile [\[\]](#)
 abstract selectables [\[\]](#)
 custom checkout [\[\]](#)
 disconnecting [\[\]](#)
 history [\[\]](#)
 selectables [\[\]](#)
vcfolder [\[\]](#)
 custom checkout [\[\]](#)
 history [\[\]](#)
vcmodule [\[\]](#)
 abstract selectable [\[\]](#)
 history [\[\]](#)
viewFormBased parameter [\[\]](#) [\[\]](#)

W

Web Services
 authentication [\[\]](#)
 deployment [\[\]](#)
 development [\[\]](#)
 environment variables [\[\]](#)
 handling data [\[\]](#)
 marking JPOs [\[\]](#)
 scoped services [\[\]](#)
Web site
 accessing from [\[\]](#)
Window Height setting [\[\]](#) [\[\]](#) [\[\]](#)
Window Width setting [\[\]](#) [\[\]](#) [\[\]](#)
web form [\[\]](#)
webform

configurable search 

width setting 

workflow macros 

workflow

selectable fields 

