Studio Modeling Platform™

Business Modeler Guide

3DEXPERIENCE R2015x

This page specifies the patents, trademarks, copyrights, and restricted rights for the 3DEXPERIENCE Platform R2015x:

Patents

The 3DEXPERIENCE Platform R2015x is protected by one or more U.S. Patents number 5,615,321; 5,774,111; 5,844,562; 5,844,566; 5,920,491; 6,044,210; 6,233,351; 6,292,190; 6,360,357; 6,396,522; 6,396,522; 6,396,522; 6,459,441; 6,459,441; 6,459,441; 6,499,040; 6,499,040; 6,499,040; 6,545,680; 6,573,896; 6,573,896; 6,573,896; 6,597,382; 6,597,382; 6,597,382; 6,654,011; 6,654,027; 6,697,770; 6,717,597; 6,745,100; 6,762,778; 6,762,778; 6,828,974; 6,828,974; 6,904,392; 6,918,095; 6,934,709; 6,993,461; 6,993,461; 6,993,461; 7,003,363; 7,016,821; 7,152,064; 7,250,947; 7,272,541; 7,289,117; 7,400,323; 7,428,728; 7,495,662; 7,499,845; 7,542,603; 7,555,498; 7,555,498; 7,587,303; 7,587,303; 7,595,799; 7,613,594; 7,620,638; 7,676,765; 7,676,765; 7,710,420; 7,814,429; 7,814,429; 7,814,429; 7,814,429; 7,873,237; 7,913,190; 7,952,575; 7,973,788; 8,010,501; 8,013,854; 8,095,229; 8,095,886; 8,222,581; 8,222,581; 8,248,407; 8,301,420; 8,368,568; 8,386,961; 8,421,798; 8,473,258; 8,473,259; 8,473,524; 8,554,521; 8,645,107; 8,670,957; 8,686,997; 8,694,284; 8,798,975; 8,798,975; 8,812,272; 8,825,450; 8,831,926; 8,832,551; 8,847,947; 8,854,367; 8,868,380; 8,878,841; 8,89,6598; 8,907,947; other patents pending.

Trademarks

3DEXPERIENCE, the Compass icon, the 3DS logo, CATIA, SOLIDWORKS , ENOVIA, DELMIA, SIMULIA, GEOVIA, EXALEAD, 3D VIA, BIOVIA, NETVIBES, 3DSWYM and 3DEXCITE, are commercial trademarks or registered trademarks of Dassault Systèmes or its subsidiaries in the United States and/or other countries. Use of any Dassault Systèmes or its subsidiaries trademarks is subject to their express written approval.

Other company, product, and service names may be trademarks or service marks of their respective owners.

Copyright

Certain portions of the 3DEXPERIENCE Platform R2015x contain elements subject to copyright owned by the following entities:

© Modelon AB
© Distrim
© NVIDIA ARC
(c) Copyright IBM Corporation 2007 All Rights Reserved" + cf http://www-03.ibm.com/software/sla/sladb.nsf/c7134e107cf0624e86256738007531d7/fd6322f964805a37002573a70058ab2e?OpenDocument
© DLR (Deutsches Zentrum f¸r Luft und Raumfahrt)
© DISTENE
© Kjell Gustafsson
© 2000 Geometric Limited
© Weber-Moewius, D-Siegen
© Oracle
© INRIA
© INRIA
© ITI (International Technegroup Corporation)
Raster Imaging Technology copyrighted by Snowbound Software 1996-2000
© Arsenal

© Allegorithmic
Copyright (c) 1997-2004 Lattice Technology, Inc. All Rights reserved
© NVIDIA
© Intel Corp
© Regents of the University of Minnesota
© Third Millenium Productions
© Scapos
© NuoDB
ìCAM-POST Æ Version 2001/14.0 © ICAM Technologies Corporation 1984-2001. All rights reservedî
© CENIT
© IMS
Contains copyrighted materials of MachineWorks Design Limited (c) 1995-2000
© NCCS
© Copyright 2003 - 2009 by Technia AB. All rights reserved. This product includes software developped by Technia AB (http://www.technia.com)
Unpublished - Copyright 1991-1997 Synopsys Inc. All rights reserved. The program and information contained herein are licensed only pursuant to a license agreement that contains use, reverse engineering, disclosure and other restrictions; accordingly, it is "Unpublished – rights reserved under the copyright laws of the United States" for purposes of the FARs. RESTRICTED RIGHTS LEGEND: Notwithstanding any other lease or license agreement that may pertain to, or accompany the delivery of, this LICENSED PRODUCT, the rights of the Government regarding its use, reproduction and disclosure are as set forth in the Government contract.
XMLmind XSL-FO Converter Copyright © 2002-2009 Pixware SARL
Copyright Sun Microsystems, Inc.

This software is based in part on the work of the independent JPEG Group.

The **3D**EXPERIENCE Platform R2015x may include open source software components or free programs (together “OS Programs”). Each such program is distributed with **3D**EXPERIENCE Platform R2015x in binary form and, except as permitted by the applicable license, without modification. Each such program is available online for free downloading and, if required by the applicable OS Program license, the source code will be made available by Dassault Systèmes upon request.

IP Asset Name	IP Asset Version	Copyright notice
Under Academic Free License		
JAVA SWING DATE PICKER	v0.99-2006.09.01	
Under Apache 1.1		
Element Construction Set	1.4.2	Copyright (c) 1999-2003 The Apache Software Foundation. All rights reserved
Jakarta	2.07	Copyright 1999–2004, The Apache Software Foundation
JAKARTA Regular Expression	1.3	Copyright (c) 1999-2003 The Apache Software Foundation. All rights reserved.
JavaMail / Servlet-API	1.4.2 / 2.3	Copyright (c) 1999 The Apache Software Foundation. All rights reserved. VOIR AVEC R&D S IL S AGIT DU MEME COMPOSANT OU DEUX COMPOSANTS FONCTIONNANT ENSEMBLE

IP Asset Name	IP Asset Version	Copyright notice
Xalan	2.3.1	Copyright (c) 1999-2003 The Apache Software Foundation
XML4C	2.4	Copyright (C) 1998-2008, International Business Machines Corporation * and others
Code Generation Library (cglib)	2.2.2	This product includes software developed by Yale University
Under Apache 2.0		
ActiveMQ-activeIO-core	3.0.0	
Amazon Java SDK	1.3.26	
Ant	1.6.1	The Apache License Version 2.0 applies to all releases of Apache Ant starting with Ant 1.6.1
Apache Common Lang	2.0	Copyright 2001-2014 The Apache Software Foundation
Apache Commons	1.8	Copyright 2001-2012 The Apache Software Foundation
Apache Commons-cli	1.2	Copyright 2001-2009 The Apache Software Foundation
Apache Commons-codec	1.4	Copyright 2002-2013 The Apache Software Foundation
Apache Commons-Compress	1.8	Copyright 2002-2014 The Apache Software Foundation
Apache Commons-FileUpload	1.2.2	Copyright 2002-2010 The Apache Software Foundation
Apache Commons-httpclient	3.1	Copyright 1999-2011 The Apache Software Foundation
Apache Commons-io	2.1	Copyright 2002-2014 The Apache Software Foundation
Apache Commons-JEXL	1.1	Copyright 2006 The Apache Software Foundation
Apache Commons-lang		Copyright 2001-2014 The Apache Software Foundation
Apache Commons-logging	1.1.1	Copyright 2003-2013 The Apache Software Foundation
Apache HTTP Server	2.2.23	Copyright (c) 2011 The Apache Software Foundation.
Apache log4j	1.2.16	Copyright 2007 The Apache Software Foundation
Apache POI	2.5.1	Copyright 2002-2004 The Apache Software Foundation
Apache Storm	0.8	Copyright 2014 The Apache Software Foundation
Apache Tomcat	6	Copyright 1999-2014 The Apache Software Foundation
Apache.commons.fileupload	1.2.1	Copyright 2002-2008 The Apache Software Foundation
ApacheSSL	2.2.21	
Axis	1.4	Copyright 2001-2004 The Apache Software Foundation
Bean Validation API	1.0.0.GA	Copyright (c) Red Hat, Inc., Emmanuel Bernard
BoneCP	0.7.1.RELEASE	Copyright 2010 Wallace Wadge
CAS client (java)	2.1	Copyright 2010, JA-SIG, Inc
Commons Math Bundle	1.2	Minpack Copyright Notice (1999) University of Chicago. All rights reserved / This product includes software developed by the University of Chicago, as Operator of Argonne National Laboratory
Daisydiff	1.1	Copyright 2007 © Guy Van den Broeck <guy@guyvdb.eu>; Daniel Dickison

IP Asset Name	IP Asset Version	Copyright notice
Derby	10.8.2.2	(C) Copyright 1997,2004 International Business Machines Corporation. All rights reserved This product includes software developed by The Apache Software Foundation (http://www.apache.org/).
Ehcache	2.4.6	Copyright 2003-2010 Terracotta, Inc.
Formatting Objects Processor (FOP)		Copyright (c) 1998-1999, James Tauber. All rights reserved.
GChart	2.3	Copyright 2007,2008,2009 John C. Gunther
Google Web Toolkit (GWT)	1.5	Copyright 2007, Google Inc.
Guava	14.0	Copyright (c) 2011 Guava Authors. All rights reserved
GWT Drag and Drop	2.5.6	Copyright 2009 Fred Sauer
GWT Incubator	1.5	Copyright 2008, Google Inc.
GWTx	1.5-20081912	Copyright 2009 Google Inc.
Hibernate Validator Engine	4.3.1.Final	Copyright 2009, Red Hat, Inc. and/or its affiliates
HTTPClient	3.1	Copyright 1999-2007 The Apache Software Foundation
ibatis-core	3.0	
Ini4j 0.5.2	0.5.2	Copyright 2005,2009 Ivan SZKIBA
Inspektr	1.0.7.GA	Copyright 2010 Rutgers, the State University of New Jersey, Virginia Tech, and Scott Battaglia
iossim		Copyright (c) 2009-2013 by Appcelerator, Inc. All Rights Reserved. / A TRANCHER AVEC R&D
Jackson	1.9.12	Copyright (c) 2007- Tatu Saloranta, tatu.saloranta@iki.fi
jakarta.commons.lang	2.4	Copyright 2001-2008 The Apache Software Foundation
Jakarta.commons.logging	1.0.1	Copyright 2001-2004 The Apache Software Foundation.
jakarta.commons.net	1.4.0	Copyright 2001-2005 The Apache Software Foundation
Jasper	5	Copyright 1999-2010 The Apache Software Foundation
Jettison	1.3.2	Copyright 2006 Envoi Solutions LLC
JNRPE		Copyright (c) 2008 Massimiliano Ziccardi
Joda Time		Copyright 2001-2012 Stephen Colebourne
json simple	1.1	Copyright ©FangYidong<fangyidong@yahoo.com.cn>
JUG (Java UUID Generator)	1.1.2	Copyright (c) 2010 Tatu Saloranta
log4j		Copyright 2010 The Apache Software Foundation
opencsv	2.0	
Tomcat	6	Copyright 1999-2014 The Apache Software Foundation
TRUEZIP	6.7 Beta 2	Copyright (C) 2009 Schlichtherle IT Services
VIAMobileIntegration	2009	Copyright 2009 Facebook
WatiN	1.3	Copyright Jeroen van Menen 2011

IP Asset Name	IP Asset Version	Copyright notice
Xalan C++	1.10	Copyright (c) 1999-2012 The Apache Software Foundation
Xerces C++	3.01	Copyright © The Apache Software Foundation
Xerces-J	2.6.2	Copyright © The Apache Software Foundation
Under Apache 2.0	Or LGPL 2.1	
Javassist	3.15.0-GA	Copyright (c) 1999-2005 Shigeru Chiba. All Rights Reserved.
Under Apache 2.0	Or BSD 2	
CardMe	0.3.6.01	Copyright 2011 George El-Haddad. All rights reserved.
Under Apache 2.0	Or BSD 3	
CAS Client (PHP)	1.3.2	Copyright 2007-2011, JA-SIG, Inc.; Copyright © 2003-2007, The ESUP-Portail consortium; Copyright (c) 2009, Regents of the University of Nebraska All rights reserved.
Under Apple License		
KeychainItemWrapper		Copyright (C) 2010 Apple Inc. All Rights Reserved.
Under ASM license		
ASM Core	3.2	Copyright (c) 2000-2011 INRIA, France Telecom
Under BeOpen Python License Agreement		
Cookie	Python-2.7	Copyright 2000 by Timothy O'Malley
Under Boost license		
Wild Magic Library		Geometric Tools, LLC // Copyright (c) 1998-2014
Boost		Copyright Joe Coder 2004 - 2006.
Or BSD 2		
yasm	1.2.0	Copyright (C) 2003-2007 Peter Johnson
xmppframework	3.2	Copyright (c) 2007, Deusty Designs, LLC
FMI Interface MAProject		Copyright The Modelica Association
Or BSD 3		
_wincon.c	2001-05-08	Copyright (c) 1999-2001 by Secret Labs AB. # Copyright (c) 1999-2001 by Fredrik Lundh.
Adaptive Simulated Annealing (ASA)	v 23.7 2001/10/12 14:01:08	Copyright © 1987-2014 Lester Ingber. All Rights Reserved.
ANTLR	1.33MR33	Copyright © 2003-2006, Terence Parr ANTLR 3 / Public domain ANTLR 2
Atmosphere & Ocean	v2	Copyright (c) 2008 INRIA
jaxen	1.1.1	Copyright 2003-2006 The Werken Company. All Rights Reserved.
JGraphX	1.3.1.6	Copyright (c) 2001-2009, JGraph Ltd
Kiss FFT	1.3.0	Copyright (c) 2003-2010 Mark Borgerding
libogg	1.2.0	Copyright (c) 2002 Xiph.org Foundation

IP Asset Name	IP Asset Version	Copyright notice
libTheora	1.1.1	Copyright (c) 2002-2009 Xiph.org Foundation
libVorbis	Vorbis I Release: 1.3.1 (Feb, 3, 2010)	Copyright (c) 2002-2008 Xiph.org Foundation
Penner's easing functions		Copyright © 2001 Robert Penner
Skia Graphics Library		Copyright (c) 2011 Google Inc. All rights reserved
V8	3	Copyright 2006-2011, the Google V8 project authors
Visualization Toolkit (VTK)	5.10	Copyright (c) 1993-2008 Ken Martin, Will Schroeder, Bill Lorensen
Vorbis	1.3.3	Copyright (c) 2002-2008 Xiph.org Foundation
vpx	1.1.0	Copyright (c) 2010 The WebM project authors
yamdi	1.8	Copyright (c) 2007-2010, Ingo Oppermann
yuicompressor	2.4.7	Copyright (c) 2013, Yahoo! Inc.
Zend Framework	1.10.2	Copyright (c) 2005-2014, Zend Technologies USA, Inc. All rights reserved.
ZipJS		Copyright (c) 2013 Gildas Lormeau. All rights reserved.
ical4J		Copyright (c) 2012, Ben Fortuna * All rights reserved.
XStream	1.3.1	Copyright (c) 2003-2006, Joe Walnes Copyright (c) 2006-2009, 2011 XStream Committers
Data Driven Documents (D3)	3.0.0	Copyright (c) 2010-2014, Michael Bostock All rights reserved.
ESAPI	2.1	Copyright © 2009 The OWASP Foundation.
GCC-XML		Copyright 2002-2012 Kitware, Inc., Insight Consortium. All rights reserved.
hsqldb	2.2.9	Copyright (c) 2001-2011, The HSQL Development Group * All rights reserved.
Under BSD Style		
itcl	3.4	This software is copyrighted by Lucent Technologies, Inc., and other parties
dom4j	1.6.1	Copyright 2001-2005 MetaStuff Ltd.. All Rights Reserved
Exodus II	4.84	Copyright (c) 2005 Sandia Corporation
Kiss FFT	1.3.0	Copyright (c) 2003-2010 Mark Borgerding All rights reserved.
FreeType	2	Copyright 2000 The FreeType Development Team
JDOM	1.0	Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved.
Natural Comparator	n.a.	Copyright (c) 2006, Stephen Kelvin Friedrich, All rights reserved. This a BSD license.
Under Castor License		
Castor	0.9.3.9	Copyright 2000 (C) Intalio Inc. All Rights Reserved.

IP Asset Name	IP Asset Version	Copyright notice
Under CDDL 1.0	Or GNU GPLV1.0	
JavaMail	1.4.2	Copyright 1997-2007 Sun Microsystems, Inc. All rights reserved.
Under CDDL 1.0	Or GNU GPL V2.0 classpath exception	
JBoss Transaction 1.1 API	1.0.0.Final	Copyright (c) 2011 Oracle and/or its affiliates. All rights reserved
Under CDDL 1.1		
Java Message Service	3.12.1.GA	Copyright (c) 2003-2010 Oracle and/or its affiliates. All rights reserved
Under Common Public License Version 0.5		
junit	3.8.1	
Under Custom Permissive License		
KeychainItemWrapper		Copyright (C) 2010 Apple Inc. All Rights Reserved
Under Custom Permissive License		
LibJPG		Thomas G. Lane, Guido Vollbeding.
Under Custom Permissive License		
LibTIFF		Copyright (c) 1988-1997 Sam Leffler Copyright (c) 1991-1997 Silicon Graphics, Inc.
Under Customized MIT License		
Tls	1.6	Copyright (C) 1997-2000 Matt Newman
Under Customized License		
Trf	2.1p2	This software is copyrighted by Andreas Kupries
Under Customized MIT License		
Tiff library	3.5.7	Copyright (c) 1988-1997 Sam Leffler Copyright (c) 1991-1997 Silicon Graphics, Inc.
Under Customized Boost license		
VRPN		Public domain until version 7.27 and then customized Boost License with credit to "The CISMM project at the University of North Carolina at Chapel Hill, supported by NIH/NCRR and NIH/NIBIB award #2P41EB002025"
Under Customized License		
libPNG		Copyright (c) 2004, 2006-2014 Glenn Randers-Pehrson depending on the asset version - to be confirmed with R&D)
Under Eclipse Public License 1.0		
AspectJ	1.6.11	Copyright (c) 1998-2001 Xerox Corporation, 2002 Palo Alto Research Center, Incorporated, 2003-2008 Contributors. All rights reserved.
Eclipse Platform	3.3.1	A VOIR AVEC LA R&D
Graphviz	none	* Copyright (c) 2011 AT&T Intellectual Property. All rights reserved

IP Asset Name	IP Asset Version	Copyright notice
Under CDDL		
jersey	1.17	Copyright (c) 2010-2011 Oracle and/or its affiliates. All rights reserved
Info-ZIP license		
Unzip (from InfoZip)	6.0	Copyright (c) 1990-2009 Info-ZIP. All rights reserved.
Zip	3.0	Copyright (c) 1990-2009 Info-ZIP. All rights reserved.
Under Jasig License for USE		
cas-client-core	3.1.6	Copyright 2007 The JA-SIG Collaborative. All rights reserved
Under LGPL		
unix ODBC	2.2.14	A revoir avec Rodolphe
GWT Beans Binding	0.2.3	* Copyright (C) 2006-2007 Sun Microsystems, Inc. All rights reserved.
GWT Mosaic	0.1.9.1	Copyright (C) 2009 Georgios J. Georgopoulos, All rights reserved.
Hibernate Commons Annotations	4.0.1.Final	Copyright (c) 2008, Red Hat Middleware LLC
Hibernate	4.1.6.Final	Copyright (c) 2009 by Red Hat Inc and/or its affiliates
WxPython	2.8	Copyright (c) 1992-2013 Julian Smart, Vadim Zeitlin, Stefan Csomor, Robert Roebling, and other members of the wxWidgets team
Under LGPL 2.1		
Hibernate	3.17.1-GA	Copyright (c) 2007, Red Hat Middleware, LLC. All rights reserved.
HTMLPurifier	4.4	Copyright 2006-2008 Edward Z. Yang
JACOB	1.14.3	Copyright (c) 1999-2004 Sourceforge JACOB Project. All rights reserved. Originator: Dan Adler (http://danadler.com)
JBOSS	6.1.0	Copyright 2011 Red Hat Inc. and/or its affiliates and other contributors as indicated by the @author tags. All rights reserved
JBoss Logging 3	3.1.0.GA	Copyright 2011 Red Hat, Inc., and individual contributors as indicated by the @author tags
JCIFS Libraries	1.2.25b	
jregistrykey	1.0	Copyright © 2001, BEQ Technologies Inc.
jfreechart		
Tiny MCE	3.4.6	Copyright 2009, Moxiecode Systems AB
Under LGPL 2.1	Or under GNU GPL V2	
FFMpeg	1.1	Copyright © 2000 Fabrice Bellard et al.
Under LGPL 2.1	Or under Eclipse public license - v 1.0	
c3p0	0.9.1.2	Copyright: (C) 2001-2007 Machinery For Change, Inc.
Under GNU LGPL 3.0		
libmcrypt		Copyright (C) 1998,1999,2000,2002 Nikos Mavroyanopoulos
Under MIT License		

IP Asset Name	IP Asset Version	Copyright notice
_random	Python-2.7	Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura, All rights reserved.
asyncore-asynchat	Python-2.7	Copyright 1996 by Sam Rushing
Bouncy Castle Libraries		Copyright (c) 2000 - 2013 The Legion of the Bouncy Castle Inc.
Com4J		Copyright (c) 2003, Kohsuke KawaguchiAll rights reserved.
Code mirror		Copyright (C) 2012 by Marijn Haverbeke <marijnh@gmail.com> and others
Credis		Copyright (c) 2009 Justin Poliey <jdp34@njit.edu> Copyright (c) 2011 Colin Mollenhour <colin@mollenhour.com>
ctypes	1.1.0	Copyright (c) 2000 - 2006 Thomas Heller
curl.js	0.7.3	Copyright (c) 2010-2013 Brian Cavalier and John Hann
Dynamic Java	1.1.5	DynamicJava - Copyright 1999 Dyade
Easysax		Copyright (c) 2012 Vopilovskii Constantine <flash.vkv@gmail.com>
Expat	Expat XML Parser-2.0.0	
Express	3.1.0	Copyright (c) 2009-2014 TJ Holowaychuk <tj@vision-media.ca>
f2c	20100827	Copyright 1990 - 1997 by AT&T, Lucent Technologies and Bellcore.
FTGL		Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz> Copyright (c) 2008 Sam Hocevar <sam@zoy.org> Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
Hammerjs	1.0.5	Copyright (C) 2013 by Jorik Tangelder (Eight Media)
Java Cup	11	Copyright 1996-1999 by Scott Hudson, Frank Flannery, C. Scott Ananian
Java Service Wrapper	3.0.2	Copyright (c) 1999, 2004 Tanuki Software
jQuery		Copyright 2014 jQuery Foundation and other contributors
jQuery Simple Context Menu	1.0	Copyright (c) 2011, Joe Walnes
libffi	Python-2.7	Copyright (c) 1996-2012 Anthony Green, Red Hat, Inc and others.
libxml	2.4.5	Copyright (C) 1998-2003 Daniel Veillard
LittleCMS		Copyright (c) 1998-2012 Marti Maria Saguer
Markdown-js	1.0	Copyright (c) 2009-2010 Dominic Baggott // Copyright (c) 2009-2010 Ash Berlin
MD5	(none)	Copyright (C) 1999 Aladdin Enterprises. All rights reserved.
OpenCL API	1.1	Copyright (c) 2008-2010 The Khronos Group Inc.
Three.js	R58	Copyright (c) 2010-2012 three.js authors
uu	Python-2.7	Copyright 1994 by Lance Ellinghouse, Modified by Jack Jansen, CWI, July 1995
when.js	1.7.1	Copyright (c) 2011 Brian Cavalier
WINP	1.14	Copyright (c) 2004-2009, Sun Microsystems, Inc., Kohsuke Kawaguchi
Winston	0.7.2	Copyright (c) 2010 Charlie Robbins

XML Xpat Parser	1.95.4	Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper Copyright (c) 2001, 2002 Expat maintainers.
jsPlumb	V6R2014x	Copyright (c) 2010 - 2013 Simon Porritt (http://jsplumb.org)
xmlrpclib	1.0.1	Copyright (c) 1999-2002 by Secret Labs AB Copyright (c) 1999-2002 by Fredrik Lundh
Under the Modelica License		
Modelica Standard Library		Copyright The Modelica Association
Under Mozilla Public License Version 1.1a		
Mozilla Rhino	1.7R2	
Extended Message boxes		Copyright (c) 2004 Michael P. Mehl. All rights reserved (Version 1.1a)
Under Ms-LPL License		
ATLSOAPInterfaces		Copyright © Microsoft Corporation
Under Python 2.2 License		
Trace	(no version)	copyright 2001, Autonomous Zones Industries, Inc., Copyright 2000, Mojam Media, Inc, Copyright 1999, Bioreason, Inc., Copyright 1995-1997, Automatrix, Inc., Copyright 1991-1995, Stichting Mathematisch Centrum
Jython	2.5.2	Copyright (c) 2007 Python Software Foundation; All Rights Reserved
Python	Python-2.5	Copyright © 2001-2014 Python Software Foundation; All Rights Reserved
Under Zlib License		
NanoXml	2.2.5	Copyright (C) 2000-2002 Marc De Scheemaecker, All Rights Reserved.
Natural Order Sort	2004-10-10 mbp	This software is copyright by Martin Pool, and made available under the same licence as zlib
ZLib		Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler
Zlib	Zlib-1.2.4	Zlib software copyright © 1995-2012 Jean-loup Gailly and Mark Adler

This clause applies to all acquisitions of Dassault Systèmes Offerings by or for the United States federal government, or by any prime contractor or subcontractor (at any tier) under any contract, grant, cooperative agreement or other activity with the federal government. The software, documentation and any other technical data provided hereunder is commercial in nature and developed solely at private expense. The Software is delivered as "Commercial Computer Software" as defined in DFARS 252.227-7014 (June 1995) or as a "Commercial Item" as defined in FAR 2.101(a) and as such is provided with only such rights as are provided in Dassault Systèmes standard commercial end user license agreement. Technical data is provided with limited rights only as provided in DFAR 252.227-7015 (Nov. 1995) or FAR 52.227-14 (June 1987), whichever is applicable. The terms and conditions of the Dassault Systèmes standard commercial end user license agreement shall pertain to the United States government's use and disclosure of this software, and shall supersede any conflicting contractual terms and conditions. If the DS standard commercial license fails to meet the United States government's needs or is inconsistent in any respect with United States Federal law, the United States government agrees to return this software, unused, to DS. The following additional statement applies only to acquisitions governed by DFARS Subpart 227.4 (October 1988): "Restricted Rights - use, duplication and disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(l)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252-227-7013 (Oct. 1988)

Table of Contents

Preface

This *Business Modeler Guide* is designed as a reference tool for Business Administrators.

Before You Begin...

Before you begin administrative work, you should review the following information.

- *Matrix Navigator Guide*

  You should become familiar with all of the information in the *Matrix Navigator Guide* to review the basic skills necessary for using Live Collaboration.

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

A Note About Examples

The examples shown throughout this manual may indicate a version that varies slightly from that shown on your screen. Most examples apply to Matrix Version 9.

What's New?

The former *Chapter 2, Working with Users*, has been moved to the online documentation the *Administration Guide : Working with Users*.

1

Introduction to Business Modeler

Introduction

The Business Modeler is the component of the 3DSEXPERIENCE Platform that the Business Administrator uses to set up the Live Collaboration schema, which represents the company’s business model. The Business Administrator is responsible for:

- Creating the definitions for the administrative objects used in the Live Collaboration database. (When Live Collaboration is first installed, the database is empty until the Business Administrator enters data.)
- Maintaining the administrative object definitions when work/system changes require an update to the database. For example, individuals may join or leave the company, change groups or roles, or change their location or phone number. When this happens, the person definition will have to change. You would not want a person to continue receiving objects or accessing objects for a project or role s/he is no longer involved in.

Administrative Objects

Administrative object definitions are maintained by the Business Administrator. In general, these definitions define users, business objects, relationships, and the characteristics and controls associated with them.

Each administrative object describes an element of your business. When a definition is modified, the change immediately affects all the business objects in the database that use it. Some changes affect only objects that use the definition after the change has been made.

Business Administrators define the following kinds of administrative objects:

Administrative Object	Description
Attribute	A characteristic or property assigned to a business object or relationship.
Type	A “kind” of business object.
Interface	A group of attributes that can be added to business objects and connections.
Relationship	A connection made between two objects.
Format	How application files will be accessed, viewed, or printed.
Person (User)	Someone who uses the database.
Group (User)	An organization of persons.
Role (User)	A job a person might have.
Association (User)	A combination of group(s) and role(s).
Policy	How a business object will be governed.
Program	A routine that performs a task, such as opening a word processor for editing text. Programs include Event Triggers, which are set to be executed based on the occurrence of a specified database event.
Property	An ad-hoc association between administrative types.
Channel	A collection of commands, used to define the contents of a Live Collaboration portal.
Portal	A collection of channels with information to display them on a Web page.
Command	The code behind a menu option or channel element.
Menu	A collection of commands to be displayed on a Web page.
Form	A customized form (associated with an object type) in which information related to an object can be presented to or edited by the user.
Rule	User, public, and owner privileges to objects.
Business Wizard	A program which provides a simplified user interface to accomplish a task.

Administrative Object	Description
Page	Object for creating and managing reusable text data (ASCII) and used by applications to display output to a standard Web browser or small LCD device.
Resource	Object for storing binary files of any type and size and used by Applications to display output to a standard Web browser or a small LCD device.
Application	A grouping of schema used in an application.

In the chapters that follow, you will learn to create and edit each of these definitions. Each chapter presents the information for an administrative object type or group of related object types. Although it is not necessary to read each chapter in order, to be an effective Business Administrator you should become familiar with all of the administrative object types.

Users defined as Administrators require access to the administrative objects for which they are responsible. You may want to assign several people as Business Administrators, each having access only to a certain set of administrative objects. For example, one Business Administrator might be in charge of user definitions; another in charge of attributes, types, and policies; and a third in charge of programs, formats, and wizards.

Administrative Object Names

Live Collaboration is designed for you to use your exact business terminology rather than cryptic words that have been modified to conform to the computer system limitations.

Live Collaboration has few restrictions on the characters used for naming administrative objects. Names are case-sensitive and spaces are allowed. You can use complete names rather than contractions, making the terminology in your system easier for people to understand. Generally, name lengths can be a maximum of 127 characters. Leading and trailing spaces are ignored.

You should avoid using characters that are programmatically significant to Navigator, MQL, and associated applications. These characters include:

```
/ \ | * ^ ( ) [ ] { } = < > # $ % & ! ? “ ; : ,’ §
```

Legal characters in XML are the tab, carriage return, line feed, and the legal graphic characters of Unicode, that is, #x9, #xA, #xD, and #x20 and above (HEX). Therefore, other characters, such as those created with the ESC key, should not be used for ANY field in Live Collaboration, including business and administrative object names, description fields, program object code, or page object content.

You should also avoid giving any administrative object a NAME that could be confused with a keyword when parsing MQL command input. For example, a Role named “All” will be misinterpreted if you try to use the role name in the definition of an access rule (or state access definition), since “all” is the keyword that allows all access rights.

Language Aliases

Language Aliases map any administrative object’s name from its stored representation into any number of languages. When presented to the user, be it human or a program, the chosen language is displayed. Likewise, any localized name, when fed into the system, will be mapped to its original name in the database.

The chosen language can be temporarily overridden such as during execution of a program, and then easily reset to the chosen language.

Business Administrators can create any number of language aliases for all administrative objects. Aliases are added using the MQL application. Refer to the *MQL Guide : Alias Command* for additional information.

Hidden Administrative Objects

Administrative objects can be marked as *hidden* so that they do not appear in their respective choosers in Matrix Navigator. A check box is included in the basics tab of each administrative object to define an object as hidden. Users who are aware of the hidden object’s existence can enter its name manually where appropriate in Matrix Navigator. Both hidden and not hidden objects are displayed in Business Modeler.

Persons, Groups, Roles, and Associations

A *person* denotes an individual human being. The concepts of *group* and *role* represent organizations and jobs. An *association* defines a union or intersection of group(s) and role(s).

Groups define units of organization and can be arranged in a hierarchy. For example, a company can have departments and each department can have sections. Each of these is represented by a group. In the case of the departments, they have the company group as their parent group and a number of sections as subgroups. Several parallel hierarchies can exist. For example, a company may be organized both in departments and projects. People belong to organizations and, in Live Collaboration, persons are assigned to groups by the Business Administrator to represent this affiliation.

A role represents the notion of the kind of job (function) that a person performs. For example, every department might have a manager and secretary. Individual persons are assigned these separate roles. Roles can be arranged in hierarchies too. For example, a role called Engineer, may have the subroles of Designer, Analyst, and Checker.

Associations can be used both to combine groups and roles, or to specify certain persons within a group or role. You could, for example, define an association consisting of all persons in the engineering, marketing and documentation departments who worked on a specific project.

Persons, groups, roles and associations are all used to define access. When groups or roles are used, Live Collaboration checks all the subgroups and subroles to look for the person attempting an action. If the person is assigned to any of these, s/he is given permission to do the action. The use of groups and roles is preferable because, as people change in the organization, the access rules do not need to change too.

Accessing Business Modeler

There are several ways to begin a session with Business Modeler. For example, if you are working on a UNIX platform, enter the following on the command line:

```
business
```

Or, if you are working on Windows, select the business icon in the 3DEXPERIENCE Platform group or from the Start menu.

The `business` command and Business icon are defined for you when the system is installed. The word “business” is equal to a series of operating system command strings that set up the Business Modeler environment.

Once the Business Modeler is active, you will see the Business browser. When you first begin a Business Modeler session, the Session Context dialog box appears. Live Collaboration knows who you are based on your *context*. Set the context and, optionally, enter a password, as described in the next section.

Users defined as Administrators require access to the administrative objects for which they are responsible. Access to Business Modeler should be restricted to a small number of Live Collaboration users. Since the definitions seriously impact the operation and usefulness of the database, you would not want everyone to have the ability to change them. You might have only one Business Administrator or several. Obviously, the larger the database and the greater the number of users, the greater the need for people to maintain it.

Two users are defined when Live Collaboration is first installed:

- **creator**—This user has full privileges for all Live Collaboration applications.
- **guest**—This user exists for people who use Navigator infrequently.

When initially using Business Modeler, you must set the context as “creator.” Then, you should add yourself as a person (Business Administrator) in Live Collaboration. You also should add a person defined as a System Administrator. (The Business and System Administrators may or may not be the same person.)

Setting the Session Context

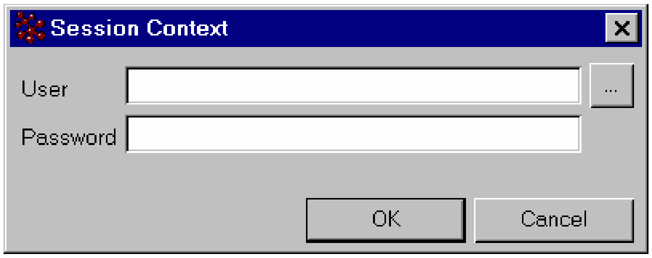
When you first begin a Business Modeler session, the Session Context dialog box appears. The Session Context dialog box lets you identify yourself (the *user*) to Live Collaboration. Context is defined using your name as defined to Live Collaboration. You may also be required to enter a password associated with your name. User name and password are case sensitive. When you identify yourself within the Session Context dialog box, you also identify what you can and cannot do while working within that Live Collaboration context.

While working in a Live Collaboration session, you can change the context using the procedure below.

To set the session context

1. Select **Context** from the Session menu.

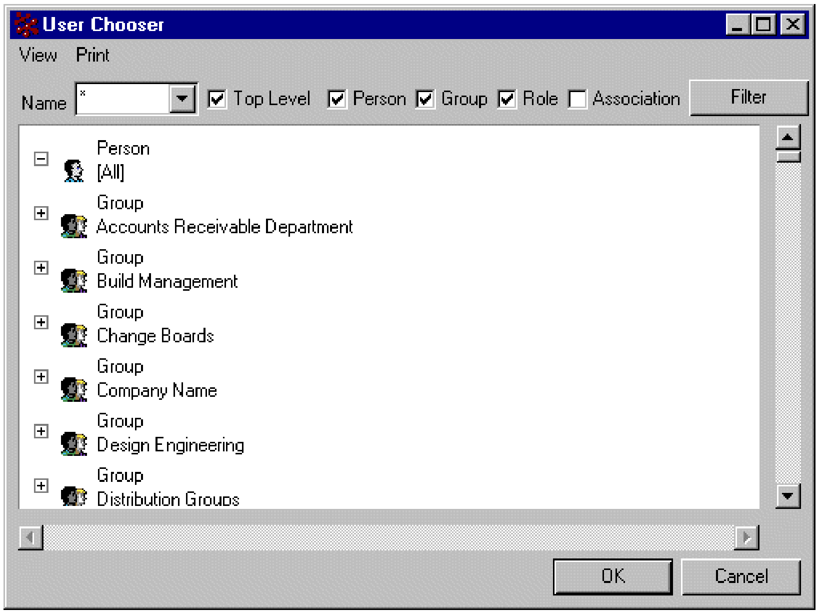
   The Session Context dialog box appears.



2. Type your user name in the **User** text box.

   Or

   Click the User **ellipsis** button and select a name from the User Chooser that appears.



   Click **OK** to confirm your choice.

3. Type a password (if required) in the **Password** text box.

4. Click **OK** to set the context and close the dialog box.

   You are now ready to work.

Context can also be set temporarily within a session via MQL.

Setting Preferences

Live Collaboration knows how you like to work with business objects based on preference settings. You can reset preferences at any time. Live Collaboration remembers preferences from one session to the next.

Application preferences are stored in the enovia.ini file. This file is stored in the user home directory (or default directory) on UNIX platforms and in the Windows directory on Windows platforms. These preferences include all options set with the Preferences option as well as most environment variables such as `font`, `menubarforeground`, `browserbackground`, and so on. A complete list of settings can be found in the *Installation Guide*.

The initialization file is read at Business Modeler startup and all visual properties of the application are set to the values in the file. When you exit Live Collaboration, the file is updated with the current settings of all options. If an initialization file does not exist, Live Collaboration creates one based on the current settings.

To set preferences

1. Select **Preferences** from the Session menu. The Session Preference dialog box appears.
2. Select your desired preferences. The preferences you can specify include:

   View by—You can specify whether business objects are displayed as icons or ImageIcons.

   Open for—You can specify whether double-clicking on a selected business object will open the object for editing or open it for viewing.
3. Click **OK** to activate your selected preferences.

Concurrent Usage

Before you work with Business Modeler features, consider that it is possible for users to actively work with the database while a Business Administrator alters it. Control should be maintained over the number of users who can act as Business Administrators, administrative changes that can take place, and when changes can occur.

You may not want to significantly change the database while users are attempting to work with it.

Communicating With Business Modeler

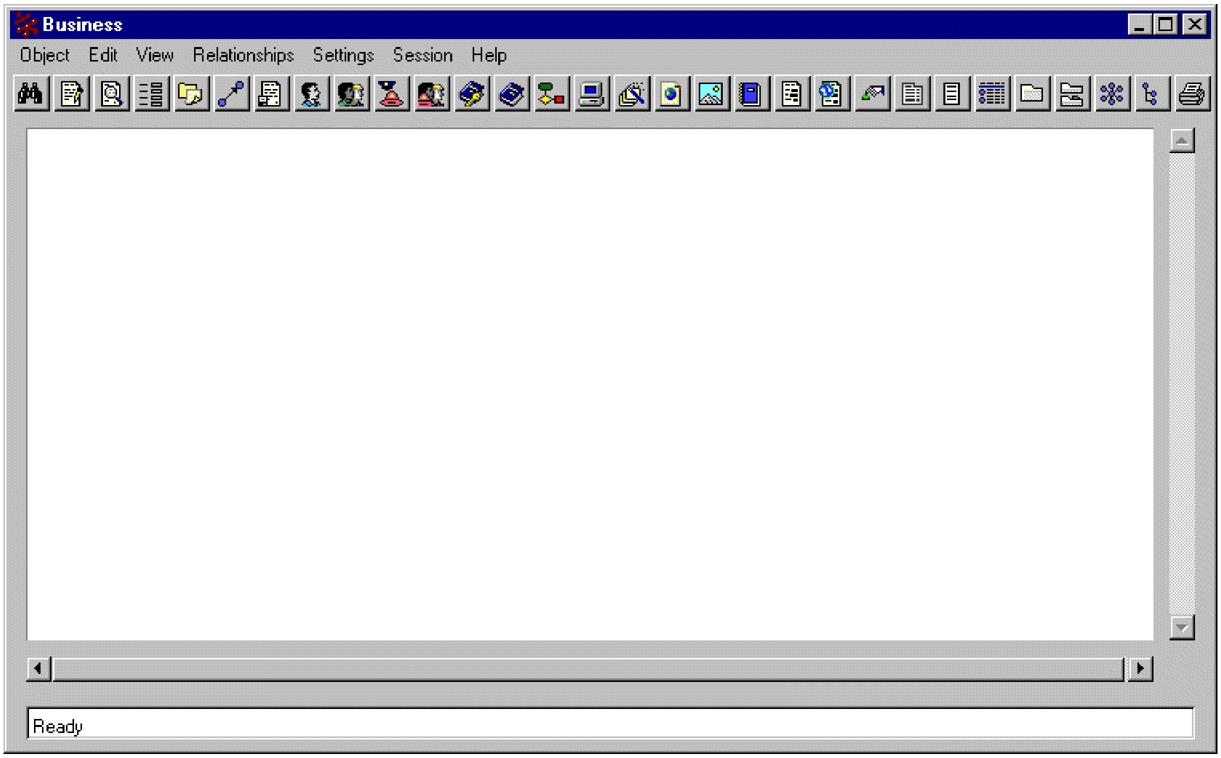
Business Modeler uses the same type of graphical interface as Matrix Navigator to display all information and request information from you.

The Business Modeler menu bar is located under the browser title. The tool bar, located just below the menu bar, provides frequently-used functions displayed in an icon form. A small pop-up window which shows the name of a tool button appears when the mouse stops on a tool button for a brief period.

The Matrix Navigator menus (with their associated options) and tools are described throughout this book.

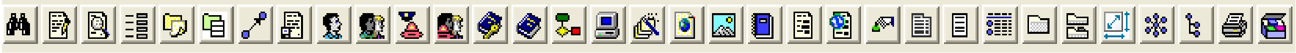
Business Modeler Primary Browser: Menus, Options, and Tools

You communicate with the Business Modeler application using the menus, options, and tools available on the Primary browser:



All menus and options are documented in the chapters that follow. The menus (with their associated options) are summarized in this section in the order in which they appear on the menu bar, from left to right:

Tools are included with associated menus options for easy referencing:



Object Menu

The Object menu offers the following options to create, view, edit, find, delete, and print administrative definitions:

Object Menu Options	
New	Creates new definitions. When you select New, another menu displays, listing the administrative definitions you can create:
	• Attribute : Adds a new attribute.

Object Menu Options	
New (Continued)	• Type: Adds a new business object type.
	• Interface : Adds a new interface.
	• Relationship: Adds a new connection type.
	• Format: Adds a new file format.
	• User Adds a new user or user category (group, role, and association): Person, Group, Role, Association.
	• Policy: Adds a new policy.
	• Rule: Adds a new rule for user access.
	• Program: Adds an external command to be executed by the system or an embedded MQL/Tcl script.
	• Wizard: Adds a new business wizard.
	• Page : Adds a new page for Web content.
	• Resource : Adds a new resource file for Web content.
	• Form: Adds a new form to be associated with object types.
	• Web Form Adds a new Web Form for use with JSP pages.
	• Command Adds a new Command for use with JSP pages.
	• Menu Adds a new Menu for use with JSP pages.
	• Inquiry Adds a new Inquiry for use with Tables.
	• Table Adds a new Table for use with JSP pages.
	• Channel Adds a new Channel for use with in Portals.
New (continued)	Portal Adds a new Portal for use with JSP pages. a new application that can be assigned to persons.

Object Menu Options	
	Application Adds a a new application.
Open	Opens an existing definition for editing or viewing. When you select Open, additional options are available: • Edit : Opens a selected definition for editing. • View : Opens a selected definition for review. You can look at the definition values but you cannot edit them.
Clone	Duplicates an existing definition.
Delete	Deletes a definition from the database. Use this option carefully!
Find	Searches the database for definitions that meet the search criteria you specify. The found definitions are displayed in the active browser.
Page Setup	Displays the system Print Setup screen where you can define printer and paper options.
Print	Displays the system Print screen where you can define the printer and number of copies.
Exit	Exits the Business Modeler application.

Edit Menu

The Edit menu lets you manipulate definitions using the following options:

Edit Menu Options	
Select All	Selects all objects in the browser.
Select None	Deselects all selected objects in the browser.

Hint: To select most of the objects in a browser choose select All from the Edit menu and then shift-click the objects to be de-selected.

View Menu

The View menu defines how objects are displayed in a browser with the following options:

View Menu Options	
Icon	Displays objects in a browser with their default icons. For example, all Person objects will be shown with the icon that is also used on the New Person button.
Image	Displays objects in a browser as ImageIcons. When an ImageIcon is not available, ENOVIA Live Collaboration displays the object as an icon. For example, each Person in the database may have an associated photo in .gif format which will be displayed when this option is selected.

Relationships Menu

The Relationships menu options enable you to choose the browser in which to display objects:

Relationships Menu Options	
Star	Displays objects in a Star browser.
Indented	Displays objects in an Indented browser.

Settings Menu

The Settings menu allows you to define system-wide settings.

Settings Menu Options	
Password	Allows you to set system-wide password settings. One setting allows you to deny access in the current session to a user who makes repeated failed login attempts. Other settings allow you to control the composition of passwords. For example, you can require that users change their passwords every 90 days, that passwords be at least six characters, and that reusing passwords be prohibited.

Session Menu

The Session menu options enable you to identify yourself as an Administrator and set preferences:

Session Menu Options	
Context	Identifies you (the user). You, the Business Administrator, define each user in Live Collaboration with a unique identifier. Each user can be a real name, badge number, or some other identification.
Preferences	Tailors the ENOVIA Live Collaboration application for your work session. You indicate that objects are displayed, by default, as icons or ImageIcons. You also indicate that a business object is opened for editing or viewing when you double click on its icon or ImageIcon. You can reset preferences at any time during a session.
Script	Creates an MQL script of all transactions performed in the session.

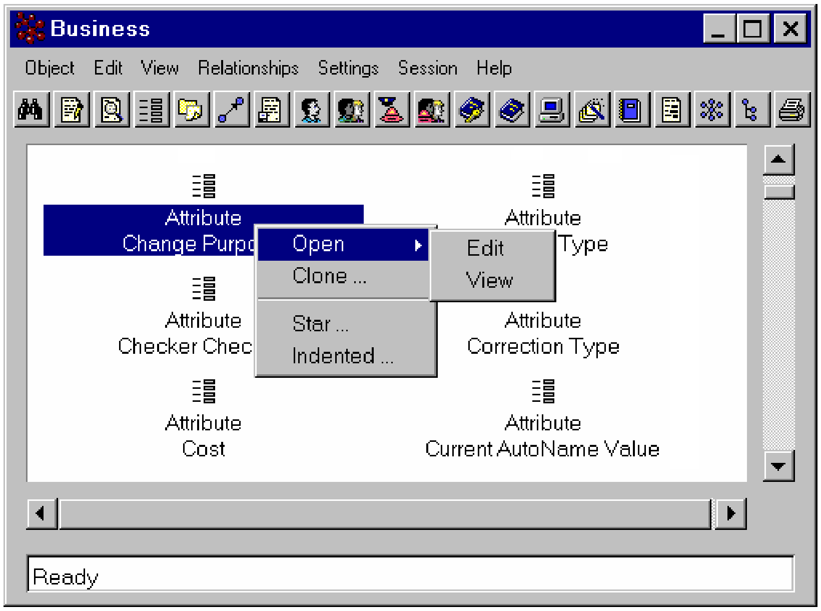
Help Menu

The Help menu option provides access to the online documents, if they have been installed. Additional Help files may also be added. Consult the *Installation Guide* for more information. The Help menu also displays information about the software version and copyright.

Help Menu Options	
On Business	Provides on-line help for the Business Modeler application by launching the help viewer.
On MQL	Provides on-line help for the MQL application by launching the help viewer.
About	Displays information about the software version and copyright.

Popup Menus

The Business Modeler Primary browser also provides access to the Open, Clone, Star and Indented menu items via a popup menu. The method to display popup menus differs from one platform to another—on most platforms including Windows, click the right mouse button; some platforms use a special MENU key. When the popup menu appears, move the mouse pointer to a menu item and click the left mouse button to make a selection.


Business
Object Edit View Relationships Settings Session Help
Attribute
Change Purpo
Open
Edit
View
Clone ...
Star ...
Indented ...
Attribute
Type
Attribute
Checker Chec
Attribute
Correction Type
Attribute
Cost
Attribute
Current AutoName Value
Ready

Defining System-Wide Password Settings

The system-wide password settings help protect your system from unauthorized access and enforce your company’s password policies. Some settings allows you to deny access to a user who makes repeated failed login attempts. Login failures, which can be consecutive, cumulative, or both, are tracked on a per person basis in the database. Other settings allow you to control the composition of passwords. For example, you can require that users change their passwords every 90 days, that passwords be at least six characters, and that reusing the old password be prohibited.

The system-wide password settings apply to:

- Every person defined in the database, except users who have either the No Password or Disable Password option selected.
- Every attempt at setting a context including when there is already a context, from the 3DEXPERIENCE Platform (Business Modeler, MQL, Navigator) or a 3DEXPERIENCE product.

If there is a context already set and there is a failure to login, the failure will be counted against the current context user and not the user trying to login.

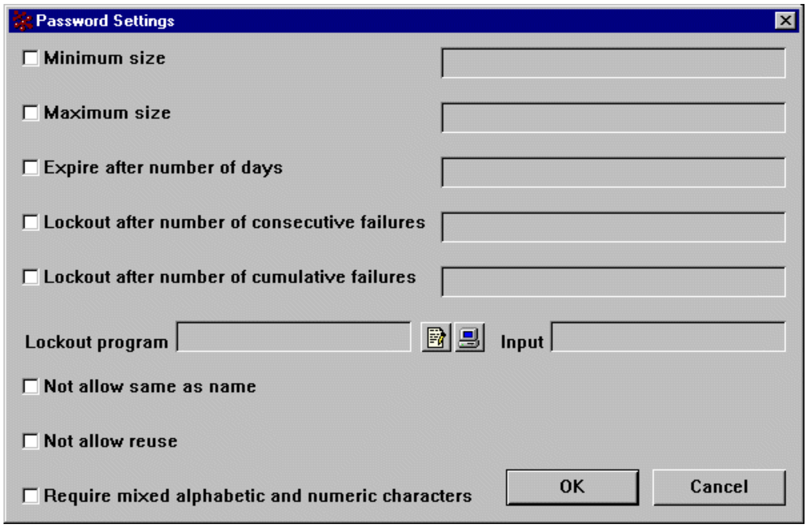
- Only passwords that are created or changed after the setting is defined, except for the expiration setting which affects all passwords. For example, suppose you set the minimum password size to 4 characters. From that point on, any password entered in a user’s person definition and all new passwords defined by the user in Matrix Navigator must be at least 4 characters. Any existing passwords that contain less than 4 characters are unaffected. (Tip: You can make passwords for existing users conform to new system-wide password settings by making users change their passwords. Do this for all users using the password expire setting or per user using the Password Change Required option.)

In a thin client environment, session is based upon session id. If you try to set the context when there is already a context, the current context is made inactivate.

To define system-wide password settings

1. Choose **Password** from the Settings menu.

   The Password Settings dialog box opens.


Password Settings
□ Minimum size
□ Maximum size
□ Expire after number of days
□ Lockout after number of consecutive failures
□ Lockout after number of cumulative failures
Lockout program
Input
□ Not allow same as name
□ Not allow reuse
□ Require mixed alphabetic and numeric characters
OK
Cancel

2. Define the settings.

   Minimum size—To require that all passwords be at least a certain number of characters, check this box and enter the number of characters. The default minimum size is 6.

   Defining a minimum password size of at least 1 ensures that users actually create a password when changing their password. If there is no minimum password size, a user could leave the new password boxes blank when changing passwords, resulting in the user having no password.

   Maximum size—To set an upper limit on the number of characters a password can contain, check this box and enter the number of characters.

The number of significant characters for password encryption and comparison can be controlled using the `cipher` *clause of the* `set password` *MQL command. See the MQL Guide : Controlling Access chapter for details.*

Lockout after consecutive / cumulative number of failures—To prevent a user from trying to login using an incorrect password *n* number of times, check one or both of these boxes and enter the number of failures allowed. Consecutive means that the count of failures is re-initialized to 0 once a successful login occurs. For cumulative, there is no re-initialization.

Since the number of failures is kept in the database, even consecutive failures can add up over multiple sessions.

After a lockout,

- It is impossible to login without restarting the process from which the login was attempted.
- The user’s person definition is changed to “inactive.” The only way for the user to log in again is to contact the Business Administrator to have the setting changed.

Lockout Program—The *Lockout Program*, fires whenever there is a failed login.

In a thin client environment, session is based upon session id. If you try to set the context when there is already a context, the current context is made inactivate.

Expire after number of days—To require that users create a new password every n number of days, check this box and enter the number of days. After the specified number of days has elapsed, the system will require users to create a new password in order to log in.

When you turn on password expiration, passwords that were created prior to version 8 will expire the next time users attempt to log in.

Not allow same as name—To prevent users from creating a password that is the same as their username, check this box.

Not allow reuse—To prevent users from entering the same password as their old password, check this box.

Require mixed alphabetic and numeric characters—To require that passwords contain at least one number and at least one letter, check this box.

Lockout Program

The lockout program fires upon any failed login. The program code can include the following macros designed to hold details allowing the logic to determine what happened.

Macro	Description
CONTEXTUSER	Contains the name of the person (if any) to whom context was set when the login attempt started. In many cases this will be null.
LOGINUSER	Contains the name of person to whom context was attempting to be set.
CONSECUTIVE_LIMIT_REACHED	Boolean value indicating if the failure resulted in the consecutive limit being reached on this login attempt.
CUMULATIVE_LIMIT_REACHED	Boolean value indicating if the failure resulted in the cumulative limit being reached on this login attempt

The program executes immediately after the lockout occurs; that is after the user is changed to inactive

Business Modeler and MQL

In addition to Business Modeler, Live Collaboration offers a scripting language called MQL. MQL features are described completely in the *MQL Guide*.

- MQL is a text-based command line interface. It uses a set of statements that help the administrator set up and test a database.
- Business Modeler lets you work through menus and options presented in browsers. Objects and other information are shown graphically.

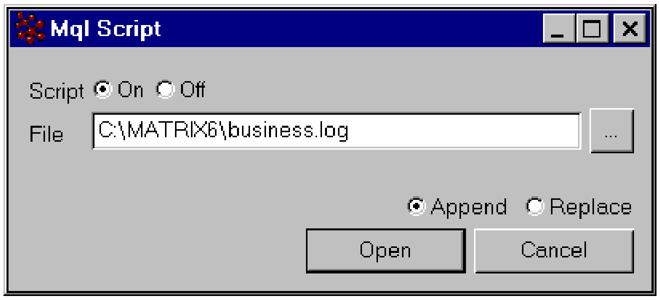
While using Business Modeler, Live Collaboration can generate an MQL script of every action that you perform. This is called *administrative MQL scripting*. The script is generated in standard system output. All changes made in a Business Modeler session can be saved in an MQL script file that you specify.

By default administrative MQL scripting is not on when you use the Business Modeler. You can turn on scripting and open a file in which to save the script.

To use MQL scripting

1. Select **Script** from the Session menu.

   The MQL Script dialog box opens:



Script ● On ○ Off

File C:\MATRIX6\business.log

● Append ○ Replace

2. Select **On** to turn on the scripting feature.
3. Type a complete path and name for the script file in the **File** text box.

   Or

   Click the **File** ellipsis button to select the path (and file) from the directories (and files) listed.

   Note that you can select a path from the list and then type a file name at the end of the path in the **File** text box.
4. Select **Append** to append current information to an existing script file.

   Or

   Select **Replace** to replace any information in the file.
5. Click **Open** to open the named file and begin to save the MQL script.

To turn off scripting

1. Select **Script** from the Session menu.
2. Select **Off** to turn off scripting.
3. Click **OK**.

Basic Functions for Creating Administrative Objects

There are some basic functions that apply to creating most definitions in the Business Modeler. They are described in the following sections.

Creating Definitions in a Specific Order

When you create definitions in the Business Modeler, the order in which you make the definitions is significant. Some definitions are dependent on previous definitions. For example, you cannot assign a person to a group if that group does not already exist. For this reason, it is recommended that you create definitions in the following order:

Definition Order		
1	Roles Groups Persons Associations *Or* Persons Groups Roles Associations	You can define roles, groups, and persons either way depending on your application. Since associations are combinations of groups and roles, they must be created after groups and roles are defined.
2	Attributes Types Relationships Formats Stores * Policies	Order is important for this set of definitions. For example, types use attributes, so attributes must be defined before types.
3	Vaults *	Vaults can be defined at any time although they are most commonly defined last.
4	Wizards Pages Resources Forms	If these items are required, they are usually defined after the database is built and tested. These definitions require workable values for reporting. If there are no values to report, it is difficult to define and test.
* Stores and vaults and are created by the System Administrator. For details, refer to the *MQL Guide*.		

Defining an Icon

You can associate a special icon to a new definition (such as a picture of a person for a person object). Icons help users locate and recognize items. When assigning an icon, it must be in GIF format. (Some GIF files are provided in the `ENOVIA_INSTALL\pixmaps\app` directory.)

To define an icon

1. Open a New definition dialog box. A New definition dialog box opens when you select a command from the Object>New menu.

For example, the New Attribute dialog box opens when you select **Attribute** from the Object>New menu. Because this dialog box would be used to define a new attribute, the default icon for an attribute is displayed in the icon area. If you were defining another object, the default icon for that object is displayed.

2. Click the **Icon** ellipsis button.

   The Icon Chooser opens.

3. Indicate the complete path to the directory containing the icon. This is accomplished by selecting the directories that define the path.

   For example, you might specify the complete path as `ENOVIA_INSTALL\pixmaps\app`.

4. Select the desired icon. Use the horizontal and vertical scroll bars as necessary.

5. Click **OK** to assign the icon.

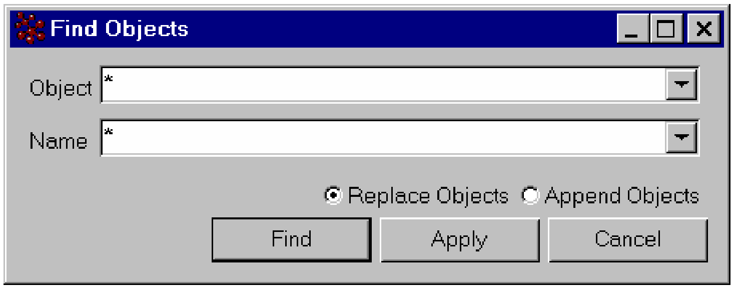
Finding Administrative Objects

The **Find** option on the Object menu searches for specific administrative objects currently defined in the database. From this list, you can select administrative objects to view, edit, and delete.

To find objects

1. Click 🔍 or select **Find** from the Object menu.

   The Find Objects dialog box opens.



Find Objects

Object: *

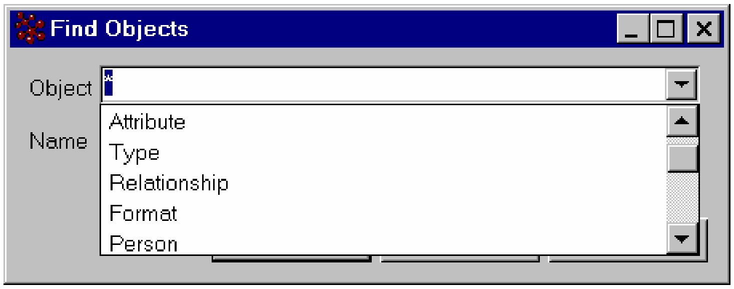
Name: *

● Replace Objects ○ Append Objects

2. Type the name of the kind of object you want to find in the **Object** text box. Remember to capitalize the first letter, for example `Attribute`.

   Or

   Click the **Object** down arrow and select the kind of object from the list.



Find Objects

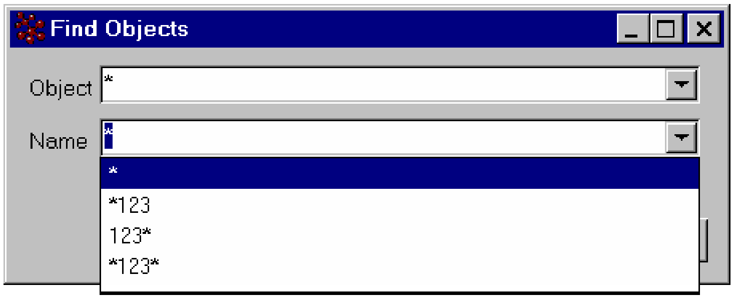
Object: *

Name

3. Type a name or partial name with a wildcard in the **Name** text box to further define the objects you want to find.

Or

Click the **Name** down arrow and select a name pattern from the list.



Object: *

Name: *

*
*123
123*
123

Replace “123” with the name pattern for which you want to search. For example, to find objects that contain the word “Property,” you could use *Propert* which will give you objects that contain the words “Property” or “Properties.” Remember that Live Collaboration is case-sensitive. You could use *ropert* if you are unsure if the initial letter is capitalized or not. The search would produce results containing “Property,” “property,” “Properties,” and “properties.”

4. Select **Append Objects** to add the found objects to the objects currently displayed in the work area.

   Or

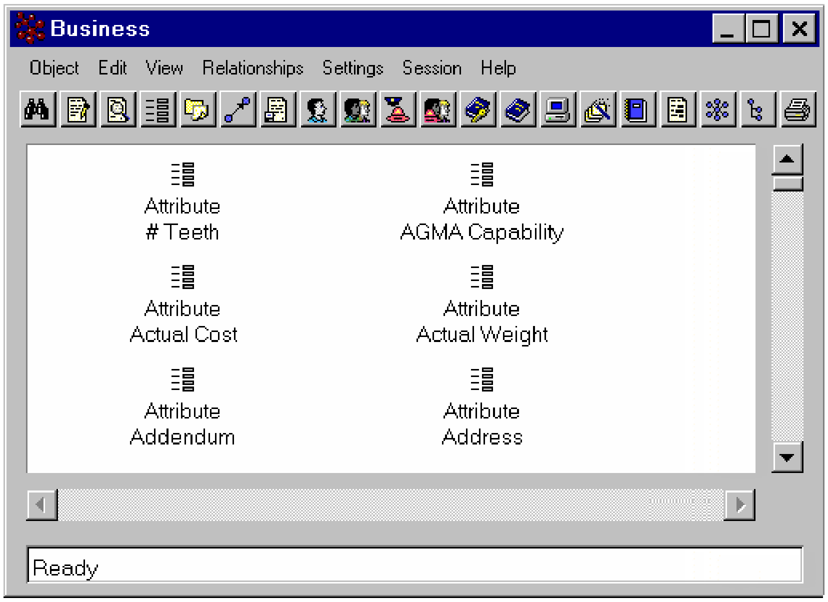
   Select **Replace Objects** to replace the current display with the found objects.

5. Click **Apply** to find the specified objects but leave the Find Objects dialog box open for a subsequent search.

   Or

   Click **Find** to find the specified objects and close the Find Objects dialog box.

   The specified administrative objects are displayed. For example, “found” attributes are shown here.



Attribute
Teeth

Attribute
AGMA Capability

Attribute
Actual Cost

Attribute
Actual Weight

Attribute
Addendum

Attribute
Address

Ready

Viewing an Administrative Object

You can view the definition of an administrative object—all of the values used to define it. Viewing administrative objects is recommended before modifying or deleting them.

To view an object definition

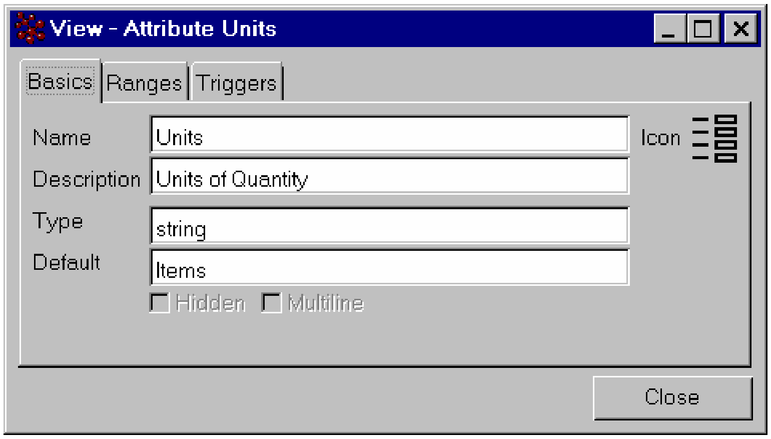
1. Select the administrative object to view.

2. Click or select **View** from the Object>Open menu.

   Or

   Move the mouse pointer over the object and click the right mouse button, then select **Open>View** from the popup menu.

   The View dialog box opens, showing the **Basics** tab. Click on the other tabs in the dialog box to show additional options. For example, the following tabs show an attribute administrative object.



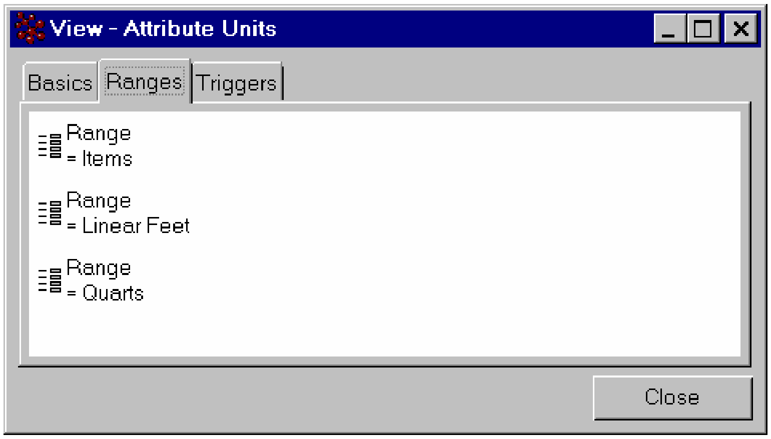
Name: Units

Description: Units of Quantity

Type: string

Default: Items

□ Hidden □ Multiline



Range = Items

Range = Linear Feet

Range = Quarts

3. After reviewing the administrative object definition, click **Close**.

Cloning an Administrative Object

You can clone an administrative object to create a duplicate object with a different name. Then, as described in *Modifying an Administrative Object*, you can modify the cloned object.

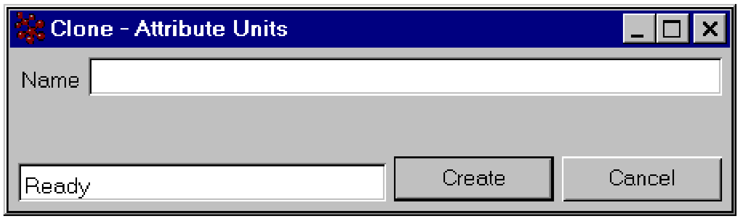
To clone an object

1. Select the administrative object to clone.
2. Select **Clone** from the Object menu.

   Or

   Move the mouse pointer over the object and click the right mouse button, then select **Clone** from the popup menu.

   The Clone dialog box opens:



3. Enter a unique name for the new administrative object.
4. Click **Create**.

   The Edit dialog box opens. It is identical to the Edit dialog box described in the chapter for the specific administrative object.
5. Edit the parameters of the object, as appropriate.

   See the chapter for the specific kind of administrative object for more information.
6. After reviewing the administrative object, click **Edit** to confirm your changes and create the clone.

Modifying an Administrative Object

After you establish an administrative object, you can add or remove defining values.

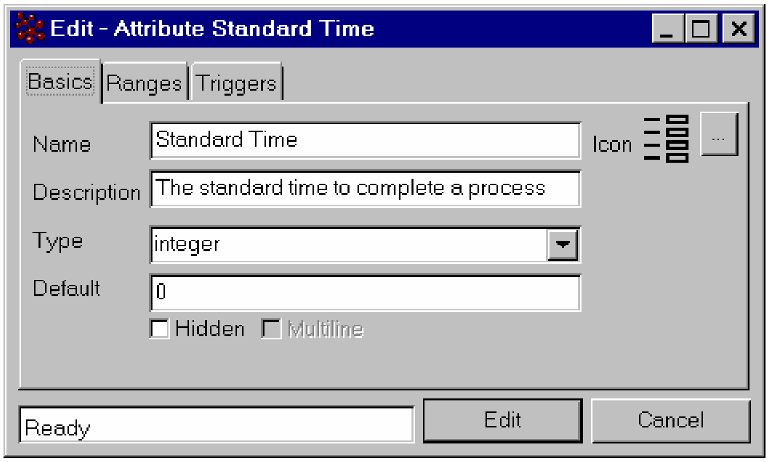
To modify an object

1. Select the administrative object to edit.
2. Double-click the object, or click 📝, or select **Edit** from the Object>Open menu.

   Or

   Move the mouse pointer over the object and click the right mouse button, then select **Open>Edit** from the popup menu.

   The Edit dialog box opens. It is nearly identical to the New administrative object dialog box. An example of an Edit dialog box for an attribute is:



Edit - Attribute Standard Time

Name: Standard Time

Description: The standard time to complete a process

Type: integer

Default: 0

□ Hidden □ Multiline

Ready

3. Edit the administrative object as appropriate. See the chapter for the specific administrative object.
4. After reviewing the administrative object, click **Edit** to confirm your changes.

Deleting an Administrative Object

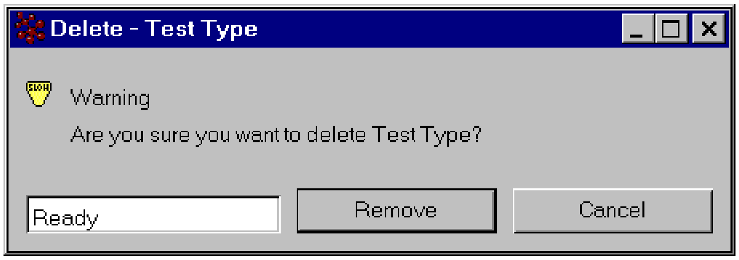
If an administrative object is no longer required, you can delete it. Note, however, that this affects all relationships and object types that use it. There may be repercussions of deleting certain administrative objects. Refer to specific chapters to determine the sort of consequences the deletion may cause.

It is recommended that you make a backup prior to deletion. Once the administrative object is deleted, you can restore its values only through a backup.

To delete an object

1. Select the administrative object to delete.
2. Select **Delete** from the Object menu.

   The Delete dialog box opens with the name and kind of selected administrative object shown in the title bar. This dialog box verifies that you want to delete the administrative object and protects you against accidental deletion.



Delete - Test Type

Warning

Are you sure you want to delete Test Type?

Ready

3. If you want to delete the administrative object, click **Remove**. If you do not want to delete it, click **Cancel**.

   If Business Objects have been created with the specified type, the following error message appears:

   ```
   Can't remove the object — Object has references
   ```

   The objects must be deleted first.

A deleted administrative object is no longer listed with the available objects.

Ending a Business Modeler Session

To end a Business Modeler session

- Select **Exit** from the Object menu.

  Live Collaboration returns you to the operating system.

2

Working With Attributes

Attributes are Assigned to Objects and Relationships

An *attribute* is any characteristic that can be assigned to an object or relationship. Objects can have attributes such as size, shape, weight, color, materials, age, texture, and so on.

Assigning Attributes to Objects

In Live Collaboration, you assign an attribute as a characteristic of a type of object. For example, assume the object is clothing. It might have attributes such as article type (for example, pants, coat, dress, shirt, and so on), size, cost, color, fabric type, and washing instructions. Now assume you are creating a new article of clothing. When you create this object, Live Collaboration prompts you to provide values for each of the assigned attributes. You might assign values such as jacket, size 10, $50, blue, wool, and dry clean.

The specific value for each attribute can be different for each object instance. However, all objects of the same type have the same attributes.

Assigning Attributes to Relationships

Like business objects, a relationship may or may not have attributes. Since a relationship defines a connection between two objects, it has attributes when there are characteristics that describe that connection.

The same attributes could apply to either the objects or the relationship. When an object requires additional values for the same attribute in different circumstances, it is easier to assign the attributes to the relationship. Also, determine whether the information has more meaning to users when it is associated with the objects or the relationship.

Defining an Attribute

There are several parameters that can be associated with an attribute. Each parameter enables you to provide information about the new attribute. While only a name is required, the other parameter values can further define the attribute and help users identify and use it.

To define an attribute

1. Click ☰ or select **Attribute** from the Object>New menu.

   The New Attribute dialog box is displayed, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

You must specify the name of the attribute you are creating. Attributes names must be unique. You should assign a name that has meaning to both you and the user. The attribute name field limit is 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

Consider the following examples:

`Unit of Length`
`Material Family`
`Color`
`Texture`

The attribute name will appear whenever the attribute is listed in a dialog box.

Assigning an Icon

You can assign a special icon to the new attribute. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1. In this case, Business Administrators are the only users who would see the icons.

Defining a Description

A description provides general information about the function of the attribute. The description also can provide guidance as to the type of value expected. If an attribute is assigned the wrong type, Live Collaboration will be unable to store the desired values. For example, with an attribute named “Units,” you might think it would store the units of measure (length, volume, etc.). The description might explain that the attribute stores numeric values only.

The description can consist of a prompt, comment, or qualifying phrase. This value can be a character string of any length. For example, if you were defining an attribute named “LABEL” you might use a description value similar to one of the following:

`The Shipping Destination Label`
`Is shipping label required?`
`Enter the shipping label number`

In the first description, the attribute might be a shipping address. In the second description, the attribute might have a value of yes, no, or unknown. In the third example, a number is required.

There is no limit to the number of characters you can include in the description. However, keep in mind that the description is displayed when the mouse pointer stops over the attribute in a chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Selecting a Type

A type is always required. It identifies the type of values the attribute will have. An attribute can assume five different types of values. When determining the attribute type, you can narrow the choices by deciding if the value is a number or a character string.

Type	Description
`String`	One or more characters. These characters can be numbers, letters, or any special symbols (such as $ % ^ * &). Although numbers can be part of a character string, you cannot perform arithmetic operations on them. To perform arithmetic operations, you need a numeric type (Integer or Real). *String attributes (as well as description fields) have a limit of 2,048 KB. If you expect to handle more data in an attribute, consider re-designing the schema to store the data in a checked-in file instead.*
`Real`	A number expressed with a decimal point (for example, 5.4321). The range of values it can assume depends on the local system architecture. Supports both 32-bit or 64-bit floating point numbers. Since real values are stored in the name format of the local system architecture, the precision and range of values varies from architecture to architecture. To obtain the exact range for your system, see your system manual.
`Integer`	A whole number whose range is defined by the local architecture. Supports all CPU architectures with the exception of signed 8-bit integers. Depending on the system architecture, an attribute with the integer type may assume a value within the following ranges: unsigned integer 8 bits 0 to 255 signed integer 16 bits -32,768 to 32,767 unsigned integer 16 bits 0 to 65,535 signed integer 32 bits -2,147,483,647 to 2,147,483,647 unsigned integer 32 bits 0 to 4,294,967,295
`Date and Time`	Date and time formats can be configured for both input and display. Consult the *Administration Guide* : *Configuring Date and Time Formats*. *For any field with a date format, even if you have a date setting in your initialization file and you input any date without a year, the date is accepted and converted to Sat Jan 01, 0000, 12:00:00 AM. Therefore, dates should be input as the full date, including the year.*
`Boolean`	A value of TRUE or FALSE

A note about Floating Point Precision/Output

For a floating point number F, the number of digits of accuracy maintained by Live Collaboration, and the number of digits printed depends on the magnitude of the absolute value of F.

Absolute Value	Accuracy	Output Possibilities
< 1.0	13 digits exact	“0.” + leading zeros
	14th digit rounded	+ maximum of 14 nonzero digits
>= 1.0	14 digits exact	no leading zeros
	15th digit rounded	+ maximum of 15 nonzero digits + 0's up to 15th digit + non-significant digits +”.0”

In the case where ABS(F) < 1.0, there are never any digits printed beyond the maximum of 14.

But for large numbers, it may be necessary to pad if the number of digits before the decimal point exceeds 15. These non-significant digits are subject to precision inaccuracies of the operating system, and can include random nonzero digits, as is demonstrated in the examples below by A digitsBig3 0.

Examples:

```
add bus A digitsBig1 0 policy A vault Standards;
add bus A digitsBig2 0 policy A vault Standards;
add bus A digitsBig3 0 policy A vault Standards;
add bus A digitsBig4 0 policy A vault Standards;
add bus A digitsSmall1 0 policy A vault Standards;
add bus A digitsSmall2 0 policy A vault Standards;
add bus A digitsSmall3 0 policy A vault Standards;
add bus A digitsSmall4 0 policy A vault Standards;

mod bus A digitsBig1 0 r1 1.234567890123459999;
mod bus A digitsBig2 0 r1 123456789.0123459999;
mod bus A digitsBig3 0 r1 1234567890123459999.0;
mod bus A digitsBig4 0 r1 1234567890123459999000000000.0;
temp query bus A digitsBig* * select attribute[r1] dump ' ';
A digitsBig1 0 1.23456789012346
A digitsBig2 0 123456789.012346
A digitsBig3 0 1234567890123460100.0
A digitsBig4 0 1234567890123460000000000000.0

mod bus A digitsSmall1 0 r1 0.1234567890123459999;
mod bus A digitsSmall2 0 r1 0.0001234567890123459999;
mod bus A digitsSmall3 0 r1 0.00000000000001234567890123459999;
mod bus A digitsSmall4 0 r1 0.000000000000000000000001234567890123459999;
temp query bus A digitsSmall* * select attribute[r1] dump ' ';
A digitsSmall1 0 0.12345678901235
A digitsSmall2 0 0.00012345678901235
A digitsSmall3 0 0.000000000000012345678901235
```

```
A digitsSmall4 0 0.0000000000000000000000012345678901235
```

Once an attribute is created with an assigned type, it cannot be changed. The user can associate only values of that type with the attribute.

If you realize that you have assigned the wrong type after the attribute is created, delete the old one and create a new one with the correct attribute type.

To assign a type to the attribute

- Type the type name in the **Type** text box.
  Or
- Select the **Type** from the drop-down list.

Defining a Default

You can define a default value for the attribute. This value is used when the user fails to provide one. You can specify any value that is of the attribute's type. The field limit is 255 characters.

When assigning a default value, the value you give must agree with the attribute type. If the attribute contains an integer value, you should not assign a string value as the default.

For example, assume you want to define an attribute called PAPER_LENGTH. Since this attribute will specify the size of a sheet of paper, you defined the type as an integer. For the default value, you might specify 11 inches or 14 inches, depending on whether standard or legal size paper is more commonly used.

As another example, assume you are defining an attribute called LABEL_COLOR. If the most common color is yellow, you might define the attribute default as yellow. If the user does not assign a value for the LABEL_COLOR, "yellow" is assigned by default.

Adding a Dimension to an Attribute

If you select a type of integer or real, the Dimension field is added to the New Attribute dialog box:

Assigning a Dimension

If the attribute is an integer or real type, you can optionally assign a dimension. The dimension associates units of measure with the attribute and provides the ability to convert from one unit to another.

To assign a dimension for the attribute

- Type the dimension name in the **Dimension** text box.
  Or
- Click the **Dimension** ellipsis button and select a dimension from the Chooser that appears.

Refer to *Working With Dimensions* for more information.

When you use dimensions, you should not use ranges. Ranges are ignored when an attribute has a dimension assigned.

Defining the Maximum Length

Attributes defined as type String can also contain a maximum length definition. The Max Length field allows you to define the maximum number of characters the attribute's value can contain. This field is very useful in enabling greater interoperability with other systems where attribute length is also constrained. The Max Length field is only available for string attributes. After the maximum length is defined, an error will occur if you attempt to create or modify an attribute with a greater number of characters.

If you modify the maximum length definition for an attribute to a number less than the current number, existing attributes will be maintained with their longer values and new attributes will need to follow the new maximum length.

By default maximum length is defined as zero (0) where zero means an unlimited number of characters. The number you define for the maximum length does not take into account the different character encoding types (UTF8, UTF16, etc.).

Defining the Reset On Option

You can determine whether an attribute value should be reset to its default value when a business object or connection is cloned or revised. An attribute's default value is defined in the Default field. A value must be defined in the Default field to use the Reset On options. Mark the **Clone** option to reset the attribute value to its default value in new business object and connection clones. Mark the **Revision** option to reset the attribute value to its default value in new business object and connection revisions.

Unmark either option to retain the current attribute value in new business object and connection clones and revisions. By default, the current attribute value is retained.

The float and replicate options for cloning and revising connections will still reset the attribute value to its default value when the appropriate option is marked.

Defining the Hidden Option

You can mark the new attribute as "hidden" so that it does not appear in the Attribute chooser or in the list of attributes for an object. You may want to use the hidden option if, for example, an object is under development or if it is intended only for your personal use. Hidden objects are accessible through MQL. In Navigator, hidden attributes are displayed only in the Object Inspector.

Defining the Multiline Option

If you have defined the **type** value as "string," you can select the format of the data entry field. Click the **Multiline** check box if you want the data entry field to consist of multiple lines. If this option is selected, the text wraps to the next line as the user types. The text box is scrollable.

If the **Multiline** option is not selected, then the data entry field consists of a single line. If the amount of text exceeds the size of the field shown, the line of text scrolls to the left as the user is typing.

Defining the Value Type

The Value Type options allow you to specify the attribute as being single-value, multi-value, or range-value. Selecting SingleValue creates a single-value attribute; selecting MultiValue creates a multi-value attribute; and selecting RangeValue creates a range-value attribute. For more information, see the *Configuration Guide: Working with Multi-Value and Range-Value Attributes*.

Defining the Scope

The Scope options allow you to specify the scope of the new attribute.

Selecting **Global** creates an attribute that applies to all types, relationships, and interfaces.

Selecting any of the other three options creates an attribute that applies only to a specific type, relationship, or interface, respectively. It also activates the **Owner** field in which you can enter the desired attribute owner, or click the ellipsis button to choose an owner using the Type Chooser.

Defining a Range

You can define the range of values the attribute can assume. A range provides a way to constrain the user's input and gives the user a way of searching a list of attribute values.

When you use dimensions, you should not use ranges. Ranges are ignored when an attribute has a dimension assigned.

If you define an attribute as having a specific range, any value the user tries to assign to that attribute is checked to determine if it is within that range. Only values within the defined range are allowed. If the range consists of a single value, only that value is allowed. Many separate range specifications can be made for an attribute.

Assume you want to restrict the user to entering only positive numbers. In this case, you could define the range using either of the following:

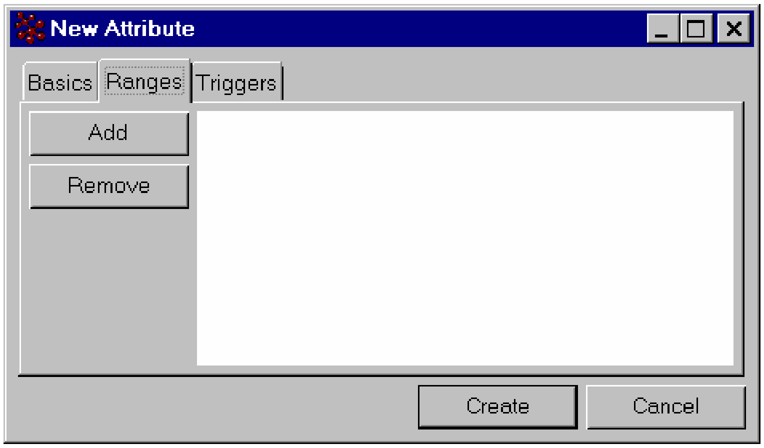
Range Statement	Is equivalent to the statement…
greater than -1	Is user's value greater than -1?,
greater than or equal 0	Is user's value greater than or equal to 0?

If the user enters a negative number (such as -1) the entry is false and, therefore, invalid (-1 is not greater than -1 and is not greater than or equal to zero).

If you have an attribute with a default value that is outside the ranges defined and you try to create a business object without changing the attribute value, Live Collaboration does not check the default value against the ranges. A trigger check could be written on the attribute to force a user to enter a value.

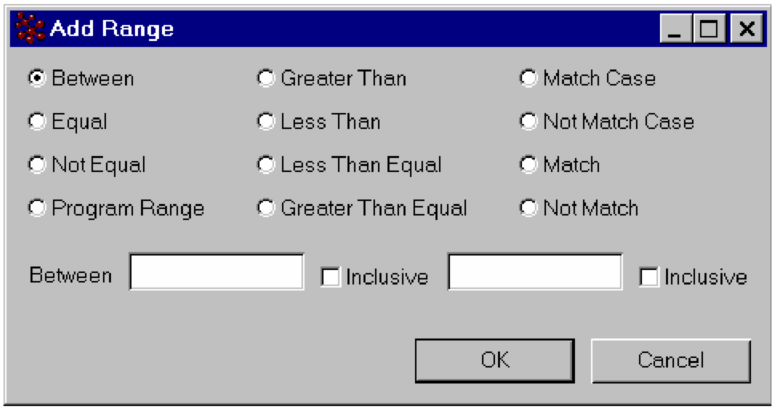
To specify a range

1. Click the **Ranges** tab in the New Attribute dialog box. The following dialog box is displayed:



2. Click **Add**.

   The Add Range dialog box is displayed.



3. Select a relational operator for the range you want to specify.

   Review the descriptions below to determine which operator to select.

   The lower portion of the Add Range dialog box changes to reflect the information you must enter based on the selected operator.

4. Enter the appropriate information and click **OK**.

Since the formats specified in the .ini file are enforced, all date attribute ranges must adhere to this format, and EVERY user must use the same setting for the display (NORMAL_FORMAT) variables, if date attributes with range values are used. It may be easiest to use the default setting for these variables. Otherwise, when you create a business object of a Type that has a date attribute with a default range specified in a different format, the default value for that date does not show up for your business object, and you will receive the following error message when you try to load attributes:

```
Date format for the DateAttribute is invalid. The correct format
is: [Gives your formats].
```

The same is true if you try to load attributes of an object that has the date attribute set in a different format than that specified in the local .ini file.

Assigning a Relational Operator

You can assign a relational operator to the range value you give. In the first two columns of operators, there are seven relational operators. Each of these operators signifies a type of comparison that will be made between the value given by the user and the range value(s).

- **Between**—Specifies that the user-supplied value must be greater than the first range value and less than the second range value (or visa versa). With this operator, two range values are required. These range values are considered the boundaries of the range.
- **Equal**—Specifies a single valid value. The attribute value must be equal to this value. This is generally used when several ranges are defined.
- **Not Equal**—Specifies that the attribute value must not match the value given in this range. When comparing user-supplied character values to the range value, uppercase and lowercase are equivalent.
- **Greater Than**—Specifies that the attribute value must be greater than the given range value.
- **Less Than**—Specifies that the attribute value must be less than the given range value.
- **Less Than Equal**—Specifies that the attribute value must be equal to or less than the given range value.
- **Greater Than Equal**—Specifies that the attribute value must be equal to or greater than the given range value.

You may have an attribute with a few commonly entered values but that can actually be any value. To provide the user with the ability to select the commonly entered values from a menu, but also allow entry of any value, you would:

- Add the ranges of **Equal** values for the common values.
- Add a range of **Not Equal** to xxxxx.

This will allow any value (except xxxxx) and also provide a list from which to choose the common values.

When defining ranges for character strings, remember that you can also perform comparisons on them. By using the ASCII values for the characters, you can determine whether a character string has a higher or lower value than another character string. For example “Boy” is less than “boy” because uppercase letters are less than lowercase letters and “5boys” is less than “Boy” because numbers are less than uppercase letters. For more information on the ASCII values, refer to an ASCII table.

But what would you do if the attribute value had a second part that was to start with the letters REV? Character patterns are available for this reason.

Using Character Patterns to Define a Range

Live Collaboration allows you to use patterns of characters to define range values. A pattern is a character string that may or may not include *wildcard* characters. Wildcard characters can be used to represent a single digit or a group of characters. This allows you to define large ranges of valid values.

For example, you can define an attribute’s range as “DR* REV*” where the asterisk (*) is a wildcard representing *any* character(s). This range allows the user to enter any value, as long as the first half begins with the letters “DR” and the second half begins with the letters “REV”.

When using a pattern to define a range, you must use a pattern operator. These operators allow comparisons between the user's value and the pattern range. Two pattern operators allow you to check the user's entry for an exact match (which includes checking for uppercase and lowercase) and all allow wildcard character comparisons.

- **Match Case**—Specifies that the character string must match the exact pattern value given, including upper and lower case. For example, "Red Robin" is not a case-sensitive match for the pattern value "re* ro*" since the upper case R's will not match the pattern's lower case r's.
- **Not Match Case**—Specifies that the character string must not match the exact pattern value. For example, if the user entered "Red" when the pattern value is "red", it would be allowed. This is because "Red" is not an exact match to "red" due to the difference in the upper and lower case r's.
- **Match**—Specifies that the character string must only match the general pattern value, independent of character case. With this operator, case is ignored so that "RED" is considered a match for "red".
- **Not Match**—Specifies that the character string must not match the general pattern value, independent of case. For example, assume the range pattern is defined as "re* ro*". If the user entered "Red Robin", the value would not be allowed, although "red ribbon" would. That is because the first value is a pattern match (regardless of case difference) and the second is not.

Entering a Range Value

You can enter a range value as a number, character string, true or false value, or date/time. The type of value you enter must match the attribute's type. Therefore, if you define the attribute as an integer, the range value must also be an integer.

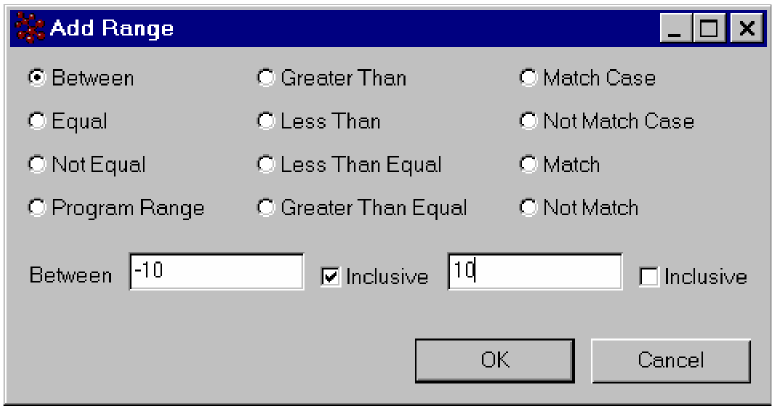
The wildcard asterisk (*) represents a group of characters. This group can have any number of characters in it. Live Collaboration will search for all values. When used with other characters, Live Collaboration will search for all values that have the characters given. For example, if you specified B* for a name search, you might get Bob, Barbara, Brenda, and Brandon. The number of characters in the name does not matter as long as the beginning letter is the letter B. If you wanted to be more specific, you could enter the value BR*. This would produce the names Brenda and Brandon since both names begin with the letters BR.

When using wildcard characters in a range value, you can insert them between other values and use them more than once. For example, you can define an attribute's range as DR* REV*. This range will allow the user to enter any value providing the first half begins with the letters DR and the second half begins with the letters REV.

Including and Excluding Values When Using the Between Operator

The **Between** operator specifies that the user's value must be within the boundaries identified by two range values. While it is clear whether the user's value is greater or less than the range values, what happens if the value is equal to one of the range values? Are the range values included as part of the range or excluded? This is specified by using the **Inclusive** option in the Add Range dialog box.

For example, assume you want to allow values between -10 and 10. You want to include -10 but not 10. You can enter the range using a single Between range:



Between -10 ☑ Inclusive 10 ☐ Inclusive

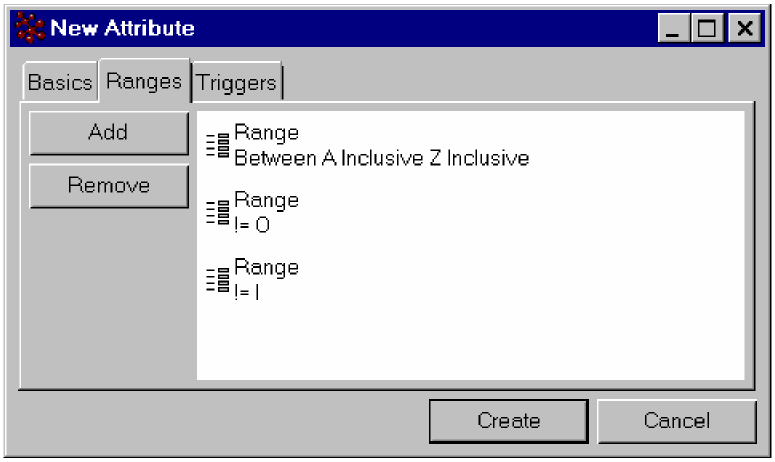
Notice that the **Inclusive** option is checked for -10. The **Inclusive** option is not checked for 10, so 10 is excluded.

Entering Multiple Ranges

You can specify more than one range. For example, assume you are defining an alphabetic range that excludes the letters I and O. This range could be represented by either of the following sets of range statements:

Between	A inclusive Z inclusive
Not Equal	I
Not Equal	O
Between	A inclusive H inclusive
Between	I exclusive O exclusive
Between	P inclusive Z inclusive

The first set of range statements would appear as:



Range
Between A Inclusive Z Inclusive

Range
!= O

Range
!= I

Using a Program to Define a Range

Live Collaboration allows you to use a program to define range values. This allows the flexibility to change range values depending on conditions. When you select the Program

Range radio button, an input field is displayed in the lower section of the dialog box where you can specify the program name to execute.

Program ranges can be used in conjunction with other defined ranges.

To define a Program Range

1. Click the **Program Range** radio button. A text box appears in the lower portion of the dialog.
2. Type the name of the program in the text box.

   Or

   Click 🖥 to display the Program Chooser and select the program.

   Only one program can be defined for any attribute.

   To edit the program named in the text box, click 📝 to display the Edit Program dialog box.
3. In the **Input** field to the right of the **Program Range** text box, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

   Environment variable “0” always holds the program name and is set automatically by the system.

   Arguments from the **Input** field are set in environment variables “1”, “2”, . . . etc.

The following is an example of a program that can be used to define ranges

```
add program nameRange
mql
  code 'tcl;
eval {
  set event [mql get env EVENT]
  set names {Larry Curly Moe}
  if { $event == "attribute choices"} {
    set output [mql get env 0]
    mql set env global $output $names
  } else {
    set value [mql get env ATTRVALUE]
    if {[lsearch -exact $names $value] == -1} {
      exit 1
    } else {
      exit 0
    }
  }
}'
;
```

Deleting a Range

To remove a range

1. In the New Attribute dialog box, click the **Ranges** tab.

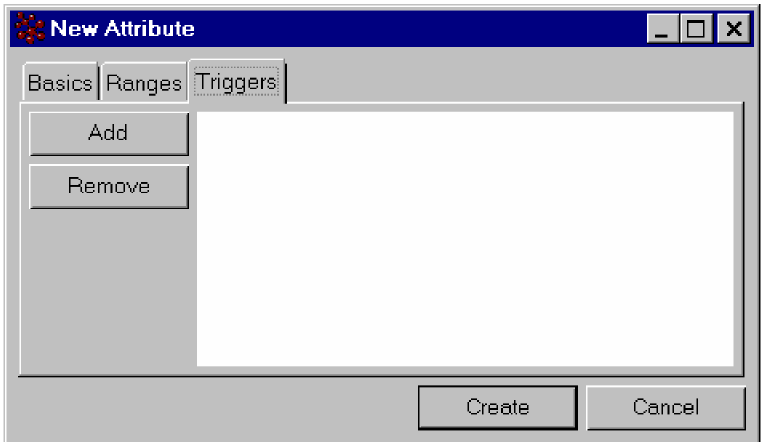
2. Select the range that you want to remove.
3. Click **Remove**.

Assigning Event Triggers

Event Triggers provide a way to customize Live Collaboration behavior through Program Objects. Triggers can contain up to three Programs—a check, an override, and an action program—which can all work together, or each work alone. Attributes support the use of triggers on the modify event. For more information, refer to the *Configuration Guide*.

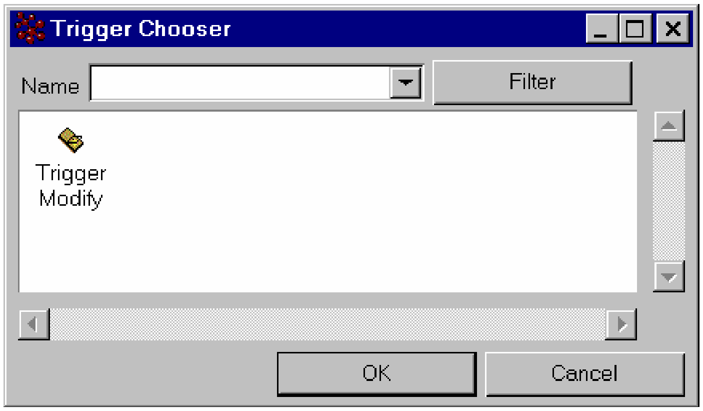
To assign event triggers

1. Click the **Triggers** tab in the New Attribute dialog box. The following dialog box is displayed:



2. Click **Add**.

   The Trigger Chooser is displayed.



3. Select the Modify trigger and click **OK**.

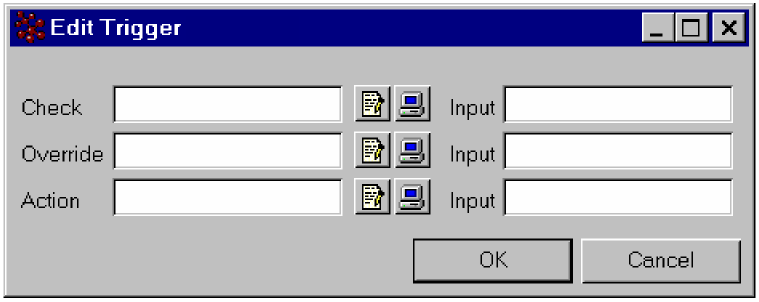
   The selected trigger is listed in the Triggers dialog.

After assigning a trigger to the attribute, you must assign the appropriate programs to each of the trigger types.

To assign programs to a trigger

1. Double-click the **Trigger** icon in the Triggers dialog.

The Edit Trigger dialog box opens.



Each supported data event has three types of triggers:

Check—a trigger that fires before the event occurs.

Override—a trigger that can replace the event transaction.

Action—a trigger that fires after the event occurs.

For the selected event, you can specify programs for none, any, or all of these trigger types. Refer to the *Configuration Guide* for more information.

2. Type the name of the program to associate with the trigger in the text box corresponding to the type of trigger.

   Or

   Click to display the Program Chooser and select the program that should execute when the event occurs.

   To edit the program named in the text box, click to display the Edit Program dialog box.

3. In the **Input** field to the right of the **Check**, **Override**, or **Action** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

   Environment variable “0” always holds the program name and is set automatically by the system.

   Arguments from the **Input** field are set in environment variables “1”, “2”, . . . etc.

4. Repeat steps 2 and 3 for each trigger type.

5. When all necessary trigger programs have been assigned, click **OK** to create the event trigger.

To remove an assigned trigger

1. In the New Attribute dialog box, click the **Triggers** tab.

2. Select the trigger that you want to remove.

3. Click **Remove**.

Creating the Attribute

After you enter all appropriate information on the New Attribute dialog box, click **Create**. A system message tells you the attribute is being created.

If you see an error message, correct the definition as required and click **Create** again.

Modifying an Attribute Definition

After you establish an attribute definition, you can add or remove defining values. Refer to the description *Modifying an Administrative Object* in Chapter 1.

Note that you cannot change the attribute's type.

3

Working With Dimensions

Overview

The dimension administrative object provides the ability to associate units of measure with an attribute, and then convert displayed values among any of the units defined for that dimension. For example, a dimension of Length could have units of centimeter, millimeter, meter, inch and foot defined. Dimensions are used only with attributes; see *Adding a Dimension to an Attribute* for instructions.

The definition of the units for a dimension includes determining which unit will be the default (the normalized unit for the dimension), and the conversion formulas from that default to the other units. The conversion formulas are based on a multiplier and offset entered when the unit is defined. The normalized unit has a multiplier of 1 and an offset of 0.

To convert to the normalized value stored in the database to a different unit, the system uses this formula:

```
normalized value = unit value * multiplier + offset
```

To display a value in units other than the normalized units, the system uses this formula:

```
unit value = (normalized value - offset) / multiplier
```

Only the normalized value is stored in the database; when an application requires the value for an attribute to be displayed, the system converts the normalized value to the to the units required. The value can be entered in any supported unit of the dimension, but it will be converted and stored in the default units.

Real attribute normalized values are stored with the same precision as real attribute values with no dimension applied. See *Working With Attributes* for more information. To avoid round-off errors with integer attributes, the default units should be the smallest unit (for example, millimeters rather than centimeters or meters).

The conversion process affects the precision. In general, up to 12 digits of precision can be assumed. For each order of magnitude that the offset and the converted value differ, another digit of precision is lost.

Dimensions help qualify attributes that quantify an object. For example, for a type with an attribute Weight, the user needs to know if the value should be in pounds or kilograms, or another dimension of weight. When the attribute definition includes a dimension, the user is provided with that information in the user interface. In addition, the user has the ability to choose the units of the dimension to enter values.

When applying dimensions to attributes that already belong to business object instantiations, refer to the *MQL Guide : Applying a Dimension to an Existing Attribute* for information about converting the existing values to the required normalized value.

Choosing the Default Units

Before you define a dimension, you need to decide which units of that dimension will be the default. That unit will have a multiplier of 1 and an offset of 0. You must calculate the multiplier and offset values for all other units of the dimension based on the default.

For example, this table shows the definition for a Temperature dimension normalized on Fahrenheit:

Unit	Label	Multiplier	Offset
Fahrenheit	degrees Fahrenheit	1	0
Celsius	degrees Celsius	1.8	32

If you wanted to normalize the dimension on Celsius, you would enter these values when defining the units:

Unit	Label	Multiplier	Offset
Fahrenheit	degrees Fahrenheit	.555555555555555	17.7777777777777777
Celsius	degrees Celsius	1	0

When you define a unit as the default, the system forces it to have a multiplier=1 and offset=0.

For dimension definitions that are to be applied to an integer, all multiplier and offset values in the dimension should be whole numbers, and the “smallest” unit should be the default.

Defining a Dimension

You can define a dimension if you are a business administrator with Attribute access. There are several parameters associated with a dimension. In addition to the Basic information, you need to define the units included in the dimension, and the multiplier and offset values used for converting to and from the normalized units.

To define a dimension

1. Click or select **Dimension** from the Object>New menu.

   The New Dimension dialog box appears, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

The dimension name field limit is 127 characters. For additional information, refer to *Administrative Object Names*.

The dimension name will appear whenever the dimension is listed in a dialog box.

Assigning an Icon

You can assign a special icon to the new dimension. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1. In this case, Business Administrators are the only users who would see the icons.

Defining a Description

A description provides general information about the function of the dimension. The description also can provide guidance as to the type of value expected.

The description can consist of a prompt, comment, or qualifying phrase. This value can be a character string of any length. For example, if you were defining a dimension named “LENGTH” you might use a description value similar to one of the following:

```
The Length with units of cm, mm, m, in, or ft,
normalized on cm.
```

There is no limit to the number of characters you can include in the description. However, keep in mind that the description is displayed when the mouse pointer stops over the dimension in a chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Defining the Hidden Option

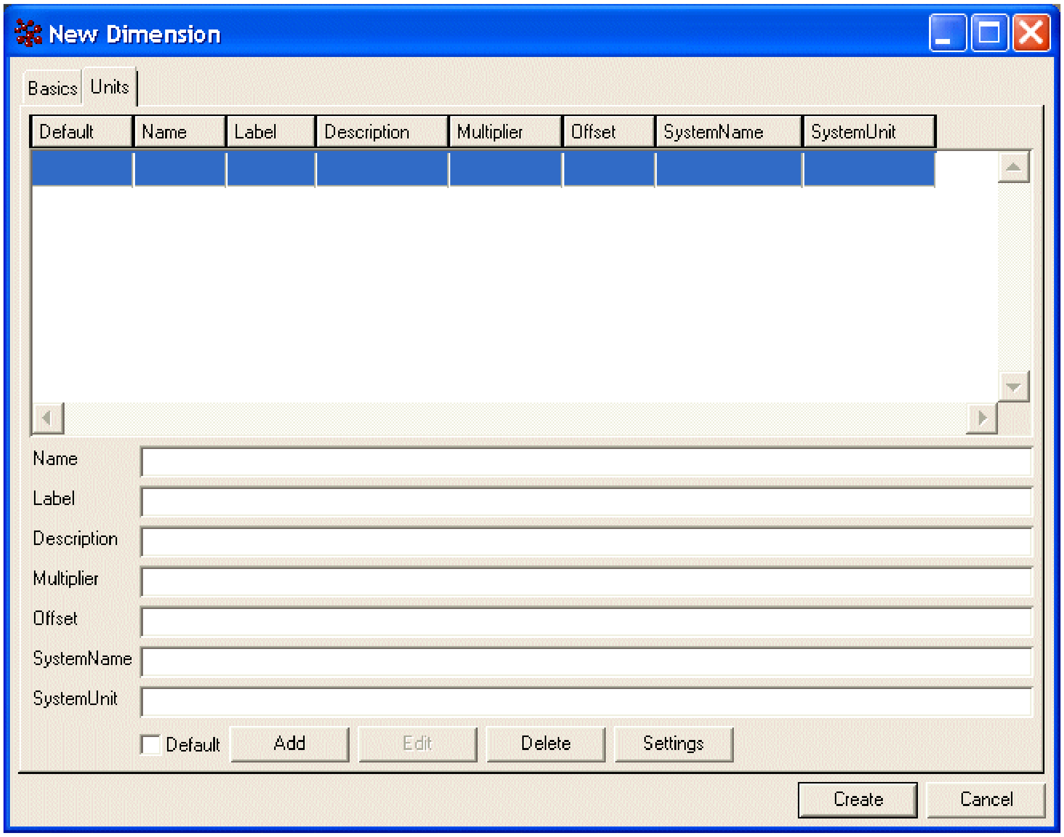
You can mark the new dimension as “hidden” so that it does not appear in the Dimension chooser. You may want to use the hidden option if, for example, an object is under development or if it is intended only for your personal use. Hidden objects are accessible through MQL.

Defining Units

For each dimension any number of units can be defined. For example, when defining a Length dimension, you may want to define units of meter, centimeter, millimeter, etc. When defining a Temperature dimension, you may want to define units of Celsius, Fahrenheit, and Kelvin.

To define units

1. Click the **Units** tab in the New Dimension dialog box. The following dialog box is displayed:



2. For each unit, enter values for these fields:

   Name—up to 127 characters

   Label—can be up to 255 characters

   Multiplier—1 for the default unit; as required for other units

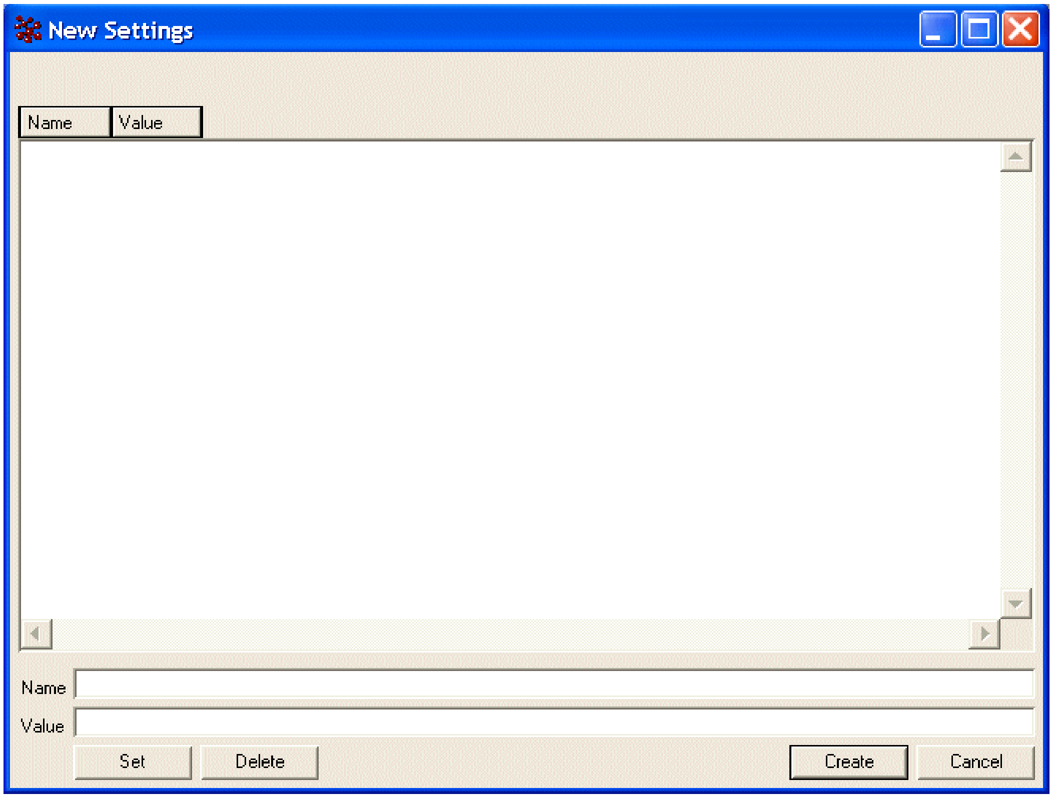
   Offset—0 for the default unit; as required for other units

   SystemName—name for a system association for this unit (such as English or Metric)

   SystemUnit—the units to convert values into when the specified SystemName is defined (for example, if converting into Metric, the units could be cm, mm, or m)

   When defining the default unit, check the **Default** check box.

3. To define settings for the dimension, click **Settings**.



a) Enter a Name and Value.

b) Click **Set**.

c) Repeat steps a and b for each setting.

d) Click **Create**. The New Settings dialog closes.

4. Click **Add**.

5. Repeat steps 2 and 3 for each unit in the dimension.

Only one unit can have a multiplier of 1 and offset of 0 (the normalized unit for the dimension). This unit is the default unit for the dimension.

Creating the Dimension

After you enter all appropriate information on the New Dimension dialog box, click **Create**. A system message tells you the dimension is being created.

If you see an error message, correct the definition as required and click **Create** again.

Modifying a Dimension

After you establish a dimension definition, you can add or remove defining values. Refer to the description *Modifying an Administrative Object*.

When modifying or deleting units associated with a dimension, you cannot modify or delete the default (normalized) units if the dimension has been applied to an attribute. You cannot remove a unit from a dimension if it has been used in any business objects as the input unit.

4

Working With Types

Overview

A *type* defines a kind of business object and the collection of attributes that characterize it. When a type is defined, it is linked to the (previously- defined) attributes that characterize it. Types are defined by the Business Administrator and are used by Live Collaboration users to create business object instances.

A type can be derived from another type. This signifies that the derived type is of the same kind as its parent. For example, a Book is a kind of Publication which in turn is a kind of Document. In this case, there may be several other types of Publications such as Newspaper, Journal, and Magazine.

This arrangement of derived types is called a *type hierarchy*. Derived types share characteristics with their parent and siblings. This is called *attribute inheritance*. Attributes, in addition to the inherited ones, can be associated with a type. For example, all Magazines have the attribute of Page Count. This attribute is shared by all Publications and perhaps by all Documents. In addition, Journals, Newspapers, and Magazines might have the attribute Publication Frequency.

Type Characteristics

Implicit and Explicit

Types use explicit and implicit characteristics:

- *Explicit characteristics* are attributes that you define and are known to Live Collaboration.
- *Implicit characteristics* are implied by the name only and are known only to the individual user.

For example, you may create a type called “Tax Form” which contains administrator-defined explicit attributes such as form number, form type, and tax year. Or, Tax Form may contain no explicit attributes at all.

When a type exists without administrator-defined attributes, it still has implicit characteristics associated with it. You would know a tax form when you saw it and would not confuse it with a type named “Health Form.” But the characteristics you use to make the judgment are implicit—known only by you and not Live Collaboration.

Inherited Properties

Types can inherit properties from other types, such as:

- *Abstract types* act as categories for other types.

  Abstract types are not used to create any actual instances of the type. They are useful only in defining characteristics that are inherited by other object types.
- *Non-abstract types* are used to create instances of business objects.

  With non-abstract types, you can create instances of the type. For example, assume that Federal Individual Tax Form is a non-abstract type. You can create object instances that contain the actual income tax forms for various individuals. One instance might be for a person named Joe Smith and another instance for Mary Jones. Both instances have the same type and characteristics although the contents are different based on the individuals.

Defining a Type

There are several parameters that can be associated with a type. Each parameter enables you to provide information about the new type. While only a name is required, the other parameter values can further define the kinds of values the type can assume, as well as provide useful information about the type.

To define a type

1. Click or select **Type** from the Object>New menu.

   The New Type dialog box opens, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Performance problems may be noticed when adding a type to a very large type hierarchy. If you experience this, from Oracle, turn off "cost-based optimization" and the situation should improve. Refer to Oracle documentation for more information.

Defining the Name

You can specify the name of the type you are creating. All types must have a unique type name assigned. You should assign a name that has meaning to both you and the user. The role name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

Consider the following examples:

Photo
Schematic
User's Manual
Insurance Form
Sales Record

The type name will appear whenever the type is listed in an Live Collaboration window.

Assigning an Icon

You can assign a special icon to the new type. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

A description provides general information about the function of the type. The description can consist of a comment or qualifying phrase. This value can be a character string of any length. However the longer the string, the more difficult it may be to read. For example, if you were defining a type named “Insurance Form” you might use a description value similar to one of the following:

```
For home owner’s insurance objects

For health insurance objects
```

In the first case, the type might contain information such as property size, property value, and previous owners. In the second case, the type might contain information about a person’s immunization history and family history of illness.

There is no limit to the number of characters you can include in the description. However, keep in mind that the description appears when the mouse pointer stops over the type in the Type Chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Assigning a Parent Type

Use the Derived From box to identify an existing type as the parent of the type you are creating. The parent type can be abstract or non-abstract. A child type inherits the following items from the parent:

- all attributes
- all methods
- all triggers
- governing policies

  For example, if two policies list the parent type as a governed type, then those two policies can also govern the child type. Note that in such a case, the child type is not listed as a governed type in the policy definitions.
- allowed types for relationships

  For example, if a relationship allows the parent type to be on the from end, then the child type can also be on the from end of the relationship. The child type is not listed in the relationship definition.

Assigning a parent type is an efficient way to define several object types that are similar because you only have to define the common items for one type, the parent, instead of defining them for each type. The child type inherits all the items listed above from the parent but you can also add attributes, programs, and methods directly to the child type. Similarly, you can assign the child type to a policy or relationship that the parent is not assigned to. Any changes you make for the parent are also applied to the child type.

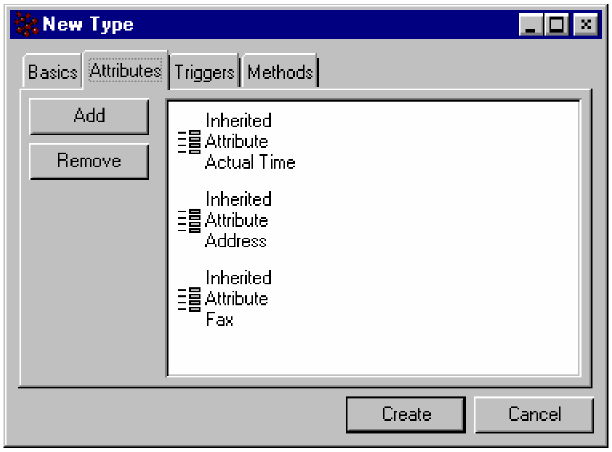
For example, suppose you have a type named “Person Record”, which includes four attributes: a person’s name, telephone number, home address, and social security number. Now, you create two new types named “Health Record” and “Employee Record.” Both new types require the attributes in the Person Record type. Rather than adding each of the four Person Record attributes to the new types, you can make Person Record the parent of the new types. The new types then inherit the four attributes, along with any methods, programs, policies, and relationships for parent.

To assign a parent type for the new type

- Type the type name in the **Derived From** text box.

  Or
- Click the **Derived From** ellipsis button and select a type from the Type Chooser that appears.

All attributes, methods, and programs added to the parent type are added to the definition for the child type. These items include the word “Inherited” so you can distinguish them from attributes, programs, and methods you add manually.



Inherited Attribute Actual Time

Inherited Attribute Address

Inherited Attribute Fax

Defining an Abstract Type

An abstract type indicates that a user will not be able to create a physical object of the type. An abstract type is helpful because you do not have to reenter groups of attributes that are often reused. If an additional field is required, it needs to be added only once.

For example, the “Person Record” object type might include a person’s name, telephone number, home address, and social security number. While it is a commonly used set of attributes, it is unlikely that this information would appear on its own. Therefore, you might want to define this object type as an abstract type.

Since this type is abstract, there will never be any actual instances made of the Person Record type. However, it can be inherited by other object types that might require the attribute information. Even though a user may never be required to enter values for the attributes of a Person Record object, he may have to enter values for these attributes for an object that inherited the Person Record attributes (such as an Employee Record or Health Record).

To define the type as abstract

- Check the **Abstract Type** checkbox.

  Types are not abstract by default.

Defining the Hidden Option

You can specify that the new type is “hidden” so that it does not appear in the Type chooser or in any dialogs that list types in Live Collaboration.

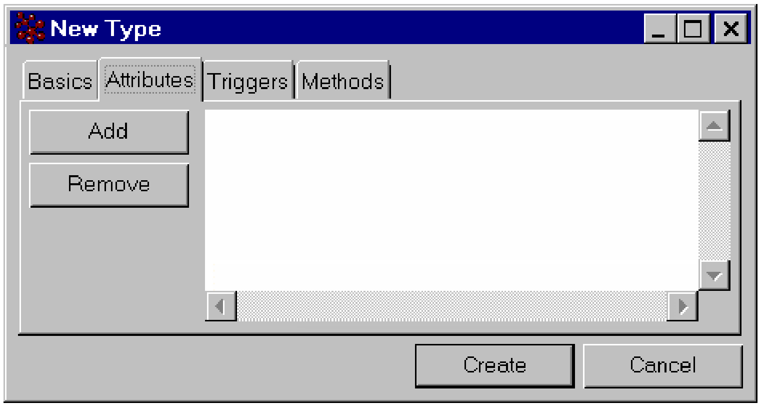
In the Studio Modeling Platform, business objects whose type is hidden are displayed based on the MX_SHOW_HIDDEN_TYPE_OBJECTS setting in the applicable ini file. Refer to the *3DEXPERIENCE Platform Administration Guide* for more information.

Hidden objects are always accessible through MQL.

Selecting Additional Attributes

You can assign to the type additional attributes from among all defined attributes available within the database. A type does not require any assigned attributes.

To select additional attributes, click the **Attributes** tab in the New Type dialog box. The following dialog box opens:



To assign attributes to the type

1. Click **Add**.

   The Attribute Chooser opens.

2. Select one or more attributes in the dialog box. Use Ctrl-click to select multiple attributes.

3. Click **OK**.

   The selected attributes are listed in the **Attributes** dialog box.

If you select an attribute that is a duplicate of an attribute assigned through inheritance, the duplication is ignored. Live Collaboration recognizes that the attributes are the same.

If you add an attribute that is part of an index, the index is disabled. Refer to the MQL Guide Index Command for more information.

To remove an assigned attribute

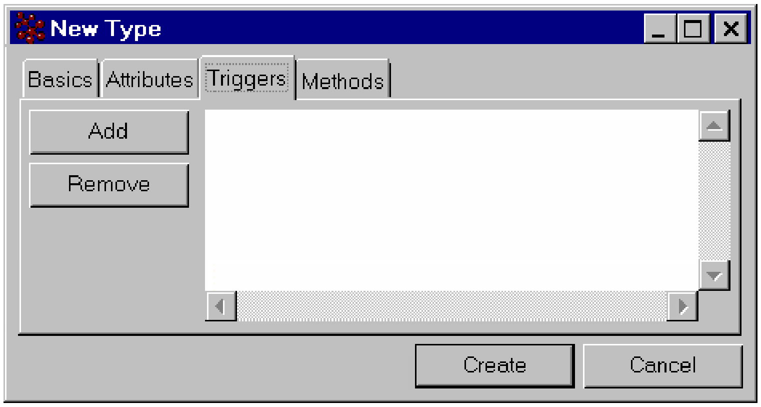
1. In the New Type dialog box, click the **Attributes** tab.
2. Select the attribute you want to remove.
3. Click **Remove**.

You can *view or edit an assigned attribute by double-clicking the attribute icon. For information on editing an attribute, see Defining an Attribute in Chapter 2.*

Assigning Event Triggers

Event Triggers provide a way to customize Live Collaboration behavior through Program Objects. Triggers can contain up to three Programs—a check, an override, and an action program—which can all work together, or each work alone. Types support triggers for many events. For more information, refer to the *Configuration Guide: About Triggers*.

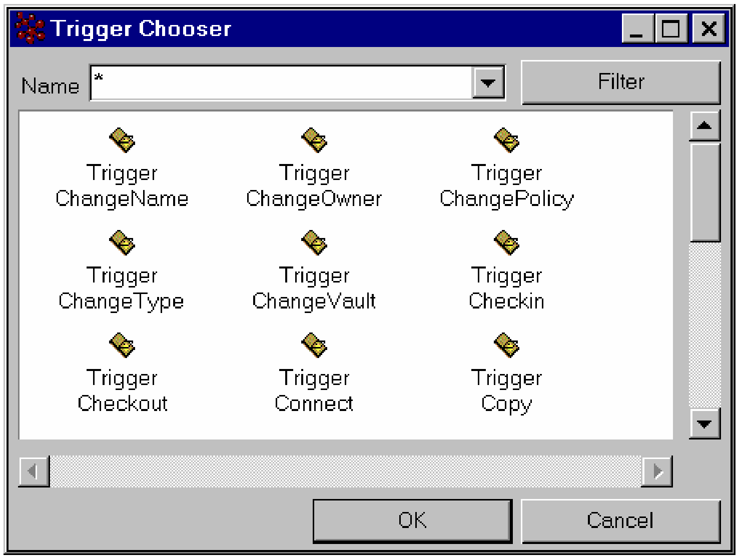
To assign event triggers, click the **Triggers** tab in the New Type dialog box. The following dialog box opens:



To assign event triggers

1. Click **Add**.

   The Trigger Chooser opens.



   The Trigger Chooser contains Trigger Objects for each of the legal events available for Types.

2. Select one or more triggers and click **OK**. Use Ctrl-click to select multiple triggers.

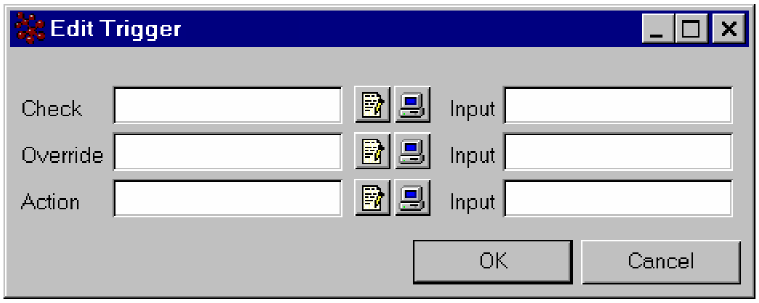
   The selected triggers are listed in the **Triggers** dialog box.

After assigning a trigger to the type, you must assign the appropriate programs to each of the trigger types.

To assign programs to a trigger

1. Double-click the trigger icon.

   The Edit Trigger dialog box opens.



Each supported data event has three types of triggers:

Check—a trigger that fires before the event occurs.

Override—a trigger that can replace the event transaction.

Action—a trigger that fires after the event occurs.

For the selected event, you can specify programs for none, any, or all of these trigger types. Refer to the *Configuration Guide* for more information.

2. Type the name of the program to associate with the trigger in the text box corresponding to the type of trigger.

   Or

   Click 🖥 to display the Program Chooser and select the program that should execute when the event occurs.

   To edit the program named in the text box, click 📝 to display the Edit Program dialog box.

3. In the **Input** field to the right of the **Check**, **Override**, or **Action** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

   Environment variable "0" always holds the program name and is set automatically by the system.

   Arguments from the **Input** field are set in environment variables "1", "2", . . . etc.

4. Repeat steps 2 and 3 for each trigger type.

5. When all necessary trigger programs have been assigned, click **OK** to create the event trigger.

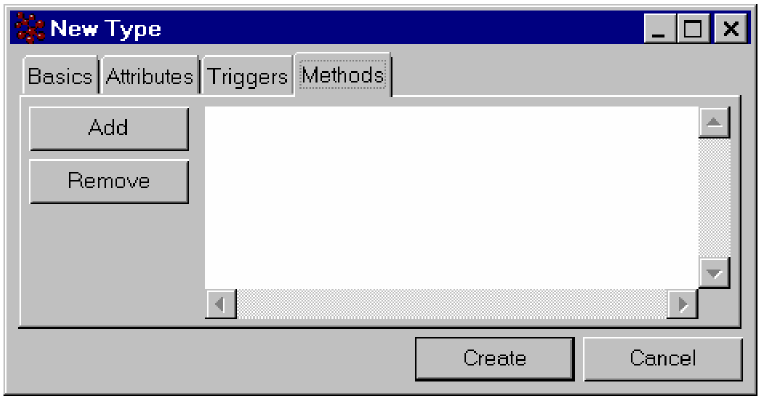
To remove an assigned trigger

1. In the New Type dialog box, click the **Triggers** tab.
2. Select the trigger you want to remove.
3. Click **Remove**.

Selecting Methods

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. Refer to *Overview of Programs* in Chapter 11 for more information.

To select methods, click the **Methods** tab in the New Type dialog box. The following dialog box opens:



To assign a method to a type

1. Click **Add**.

   The Program Chooser opens.

2. Select one or more programs in the dialog box. Use Ctrl-click to select multiple programs.

3. Click **OK**.

   The selected programs are listed in the **Methods** dialog box.

To remove an assigned method

1. In the New Type dialog box, click the **Methods** tab.
2. Select the program you want to remove.
3. Click **Remove**.

Creating the Type

After you enter all appropriate information in the New Type dialog box, click **Create**. A system message tells you the type is being created. If you see an error message, correct the definition as required and click **Create** again.

Modifying a Type Definition

After you establish a type definition, you can add or remove defining values. Refer to the description *Modifying an Administrative Object* in Chapter 1.

If the type is not inheriting attributes from a parent type, you can edit it and assign a parent. However, if the type already has a parent type, you cannot change the parent.

Two Base Types

There are two base types that must be defined in order to use some basic functions. These types are:

- Annotation
- Attachment

By creating types of these names and deriving new types from them, the number of types available under Annotation and Attachment can be kept to a manageable number. For example, when a new attachment is created, (by selecting New> Attachment from the object menu) an Attachment object type must be selected from the Type Chooser. Only the types derived from Attachment will be displayed, eliminating types that are not appropriate. The Annotation type works the same way.

Note that applicable relationships and policies for these types must be defined as well, as described in Chapter 5, *Working With Relationships*, and Chapter 9, *Working With Policies*.

Localizing Base Types

As stated above, the Annotation and Attachment types must exist in order to use these functions. However, when Live Collaboration is used in a non-English language, the non-English word for these types can be used if the following variables are set in the .ini file (the enovia.ini file for the Studio Modeling Platform or Live Collaboration Server):

> **MX_ANNOTATION_TYPE** sets the non-English word used for Annotations. A Business Object Type of the name specified here must be defined in order for the Annotation function to operate properly when used in a non-English setting. The default is Annotation.
>
> **MX_ATTACHMENT_TYPE** sets the non-English word used for Attachments. A Business Object Type of the name specified here must be defined in order for the Attachment function to operate properly when used in a non-English setting. The default is Attachment.

5

Working With Relationships

Overview

Relationship definitions are used along with the Policy to implement business practices. Therefore, they are relatively complex definitions, usually requiring some planning. Each concept mentioned in the following example is discussed in this chapter.

In manufacturing, a component may be contained in several different assemblies or subassemblies in varying quantities. In Live Collaboration, the component object can be connected to the various assembly and subassembly objects that contain it. Each time objects are connected with this relationship, the user can be prompted for the quantity value for this relationship instance. If the component is later redesigned, the older design may become obsolete. When a revision of the component object is then created, the relationship can disconnect from the original and connect to the newer revision. If the component is cloned because a similar component is available, the cloned component may or may not be part of the assembly the original component connects to. The connection to the original should remain but there should be no connection to the cloned component.

For the process to work in this fashion, the relationship definition would include the attribute “quantity.” The cardinality would be “many to many” since components could be connected to several assemblies and assemblies can contain many components. The revision rule would be “float” so new revisions would use the connections of the original. The clone rule would be “none” so the original connection remains but no connection is created for the clone.

Gathering Information

Relationships often are created through MQL at the same time that all other primary administrative objects are defined for a new system. You can also use Business Modeler to define a new relationship. Before creating a relationship definition, you must determine the following:

- Among the types of business objects that have been defined, which types will be allowed to connect directly to which other types?
- Among the types of relationships that have been defined, which types will be allowed to connect directly to which other types?
- What is the nature and, therefore, the name of each relationship?
- Relationships have two ends. The *from* end points to the *to* end. Which way should the arrow (in the Indented and Star Browsers) point for each relationship?
- What is the meaning of the relationship from the point of view of the business object on the *from* side?
- What is the meaning of the relationship from the point of view of the business object on the *to* side?
- What is the cardinality for the relationship at the *from* end? Should a business object be allowed to be on the *from* end of only one or many of this type of relationship?
- What is the cardinality for the relationship at the *to* end? Should a business object be allowed to be on the *to* end of only one or many of this type of relationship?
- When a business object at the *from* end of the relationship is revised or cloned, a new business object (similar to the original) is created. What should happen to this relationship when this occurs? The choices for revisions and clones are: `none`, `replicate`, and `float`.

  Should the relationship stay on the original and not automatically be connected to the new revision or clone? If so, pick `none`.

  Should the relationship stay on the original and automatically connect to the new revision or clone? If so, pick `replicate`.

  Should the relationship disconnect from the original and automatically connect to the new revision or clone? If so, pick `float`.
- When a business object at the *to* end of the relationship is revised or cloned, a new business object (similar to the original) is created. What should happen to this relationship when this occurs? The choices are the same as for the *from* end.
- What attributes, if any, belong on the relationship? Quantity, Units, and Effectivity are examples of attributes which logically belong on a relationship between an assembly and a component rather than on the assembly or component business object. Each instance, or use of the relationship, will have its own values for these attributes which apply to the relationship between the unique business objects it connects.

Sample Planning Template

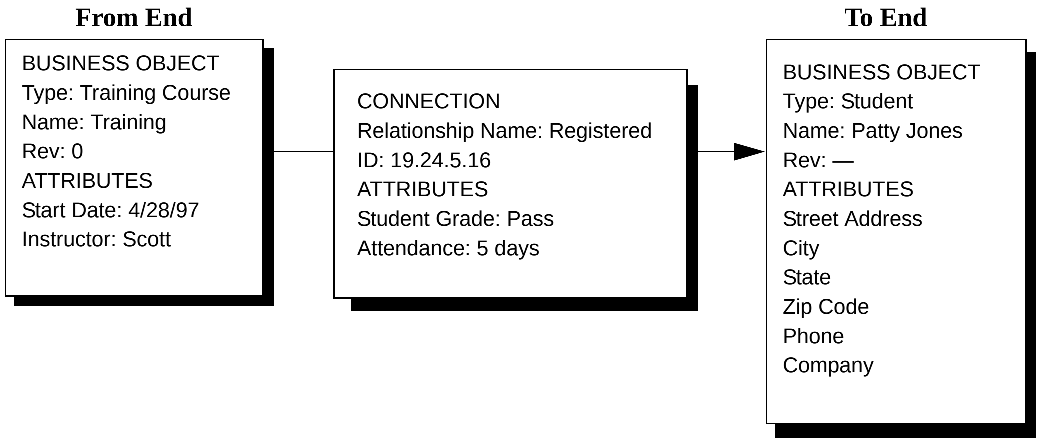
Use a table like the one below to collect the information needed for relationship definitions.

Relationship Name	________	________	________	________	________	________
From Type	________	________	________	________	________	________
From Relationship	________	________	________	________	________	________
From Meaning	________	________	________	________	________	________
From Cardinality	________	________	________	________	________	________
From Rev Behavior	________	________	________	________	________	________
From Clone Behavior	________	________	________	________	________	________
To Type	________	________	________	________	________	________
To Relationship	________	________	________	________	________	________
To Meaning	________	________	________	________	________	________
To Cardinality	________	________	________	________	________	________
To Rev Behavior	________	________	________	________	________	________
To Clone Behavior	________	________	________	________	________	________
Attributes	________	________	________	________	________	________

Relationship Definitions

When you define a relationship, you identify the types of objects and types of relationships eligible to be used by the relationship. Then you specify how each object relates to the other and provide rules for maintaining the relationship. The relationship can also have a description, icon, and attributes.

When relationships are used in Live Collaboration to connect objects, they are called *connections*. Connections are given a numeric ID that can be used to modify the attribute values in MQL.


From End
BUSINESS OBJECT
Type: Training Course
Name: Training
Rev: 0
ATTRIBUTES
Start Date: 4/28/97
Instructor: Scott
CONNECTION
Relationship Name: Registered
ID: 19.24.5.16
ATTRIBUTES
Student Grade: Pass
Attendance: 5 days
To End
BUSINESS OBJECT
Type: Student
Name: Patty Jones
Rev: —
ATTRIBUTES
Street Address
City
State
Zip Code
Phone
Company

Defining a Relationship

There are several parameters that can be associated with a relationship. Each parameter enables you to provide information about the new relationship.

To define a relationship

1. Click or select **Relationship** from the Object>New menu.

   The New Relation dialog box opens.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

You can specify the name of the relationship you are creating. All relationships must have a unique relationship name assigned. You should assign a name that has meaning to both you and the user. The relationship name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

For example, with each of the following relationship names, an implied meaning is conveyed to the user:

```
Equivalent Products
Course Evaluations
Audit Results
```

In the first example, the relationship might link two products that can be used interchangeably. The second relationship might link objects containing student reviews with a training course object. The last relationship might link an object containing a financial record with the notes from an auditor.

Assigning an Icon

You can assign a special icon to the new relationship. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

A description provides general information about the function of the relationship.

For example, assume you have two relationships named “Comment” and “Annotation.” Which relationship should you use to connect a documentation object to a drawing object? A description for each relationship should:

- Provide reasons why each relationship is defined.
- Indicate the differences between them.

For example:

`Comments connect documentation objects to projects, plans, and specifications`
`Annotations connect documentation objects to drawings and schematics`

There is no limit to the number of characters you can include in the description. However, keep in mind that the description appears when the mouse pointer stops over the relationship in a chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Defining the Derived From Option

You can specify that the new relationship is derived from an already existing relationship. Type the name of the parent relationship in the Derived From field, or click the ellipsis button … to choose an existing relationship using the Relationship Chooser. The new relationship will inherit all of the attributes, triggers, and life cycle of the parent relationship. If the parent relationship is an abstract type, check the Abstract Type box. The Meaning displayed in the From Type and To Type pages will be the same as that of the parent relationship, and the types displayed in those pages will say “Inherited Type XXX” for those types that are inherited from the parent relationship. For information about derived relationships, see the *MQL Guide : Relationships*.

Defining the Hidden Option

You can specify that the new relationship is “hidden” so that it does not appear in the Relationship chooser or in any dialogs that list relationships in Live Collaboration.

In the Studio Modeling Platform, connections whose relationship type is hidden are displayed based on the MX_SHOW_HIDDEN_TYPE_OBJECTS setting in the applicable ini file. Refer to the *3DEXPERIENCE Platform Administration Guide* for more information.

Hidden connections are always accessible through MQL.

Defining the Prevent Duplicates Option

The Prevent Duplicates check box, when enabled, prevents duplicates of the relationship type from existing between the same two objects. By default duplicates are allowed.

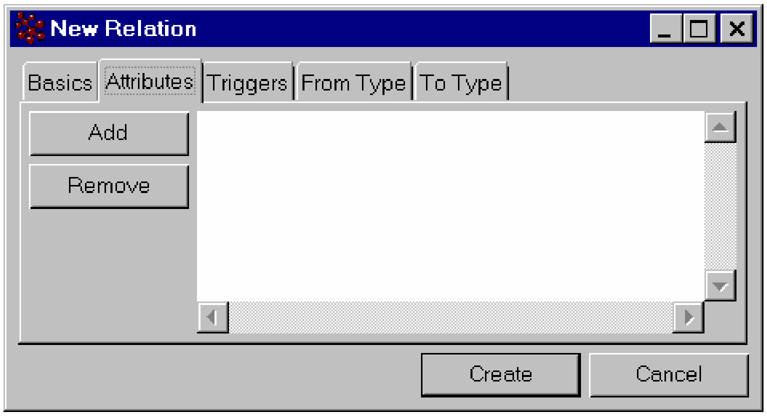
Existing relationships have this setting turned off by default.

The `preventduplicates` *flag will NOT prevent a second relationship between two objects if it points in the opposite direction. For example, given BusObjA connected ONCE to BusObjB with* `preventduplicates`*, connecting BusObjA with* `preventduplicates` *to BusObjB will fail. Connecting BusObjB with* `preventduplicates` *to BusObjA will succeed.*

Assigning Attributes

You can select attributes to assign to the relationship you are creating from among all defined attributes available within the database.

To select additional attributes, click the **Attributes** tab in the New Relation dialog box. The following dialog box opens:



To assign attributes to the type

1. Click **Add**.

   The Attribute Chooser opens.

2. Select one or more attributes in the dialog box. Use Ctrl-click to select multiple attributes.

3. Click **OK**.

   The selected attributes are listed in the Attributes dialog box.

If you add an attribute that is part of an index, the index is disabled. Refer to the MQL Guide : Index command for more information.

For example, many different Assembly objects might use a Component object. The cost for installing the component into each assembly may differ considerably. Therefore, you may want to define an attribute called “Installation Cost” and associate it with the component relationship. Whenever a connection is made between a Component object and an Assembly object, the user can insert the cost associated with that connection.

As another example, a Quantity attribute is assigned to a component object to track the number of components available. You also can assign the Quantity attribute to a relationship to track the number of components required for an Assembly. Since the quantity of the components may differ from assembly to assembly, the relationship records the amount as part of its definition. When a specific Component object is connected to a particular Assembly object, the user automatically has a means of inserting the quantity information.

To remove an assigned attribute

1. In the New Relation dialog box, click the **Attributes** tab.

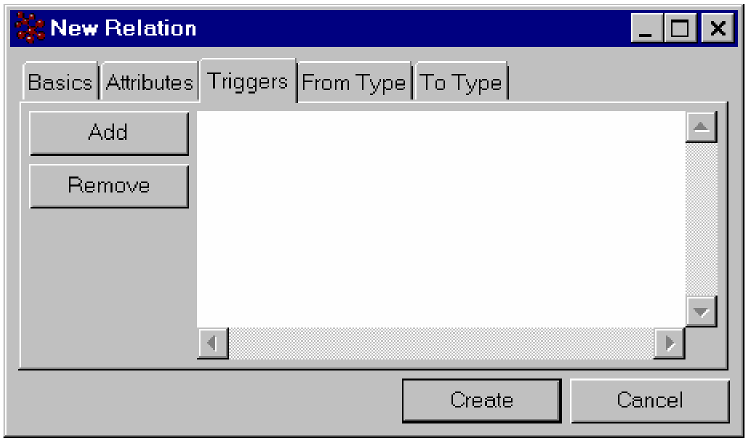
2. Select the attribute you want to remove.

3. Click **Remove**.

Assigning Event Triggers

Event Triggers provide a way to customize behavior through Program Objects. Triggers can contain up to three Programs—a check, an override, and an action program—which can all work together, or each work alone. Several Relationship events support triggers. For more information, refer to the *Configuration Guide : About Triggers*.

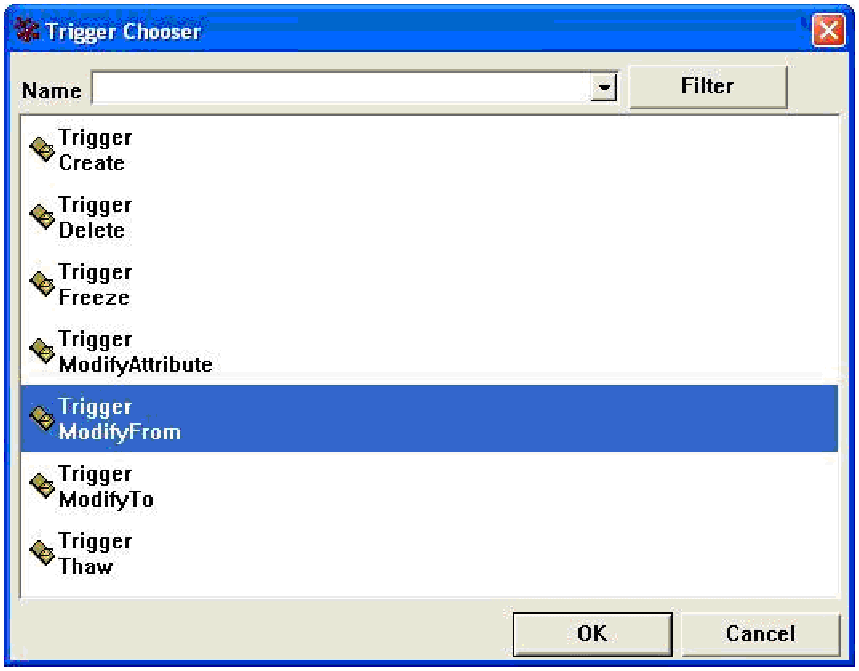
To assign event triggers, click the **Triggers** tab in the New Relation dialog box. The following dialog box opens:



To assign event triggers

1. Click **Add**.

   The Trigger Chooser opens.



   The Trigger Chooser contains Trigger Objects for each of the legal events available for Relationships.

2. Select one or more triggers and click **OK**. Use Ctrl-click to select multiple triggers.

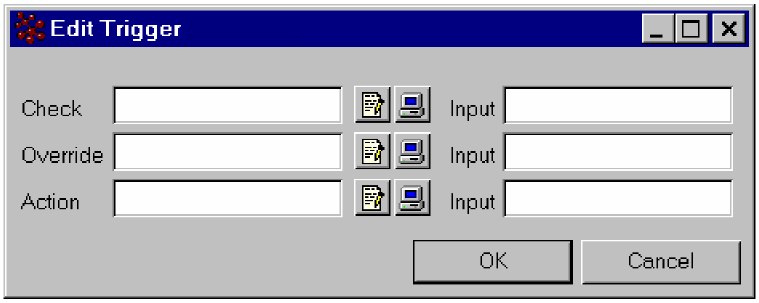
   The selected triggers are listed in the Triggers dialog box.

After assigning a trigger to the relationship, you must assign the appropriate programs to each of the trigger types.

To assign programs to a trigger

1. Double-click the trigger icon.

The Edit Trigger dialog box opens.



Each supported data event has three types of triggers:

Check—a trigger that fires before the event occurs.

Override—a trigger that can replace the event transaction.

Action—a trigger that fires after the event occurs.

For the selected event, you can specify programs for none, any, or all of these trigger types. For more information, refer to the *Configuration Guide*.

2. Type the name of the program to associate with the trigger in the text box corresponding to the type of trigger.

   Or

   Click 🖥 to display the Program Chooser and select the program that should execute when the event occurs.

   To edit the program named in the text box, click 📝 to display the Edit Program dialog box.

3. In the **Input** field to the right of the **Check**, **Override**, or **Action** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

   Environment variable “0” always holds the program name and is set automatically by the system.

   Arguments from the **Input** field are set in environment variables “1”, “2”, . . . etc.

4. Repeat steps 2 and 3 for each trigger type.

5. When all necessary trigger programs have been assigned, click **OK** to create the event trigger.

To remove an assigned trigger

1. In the New Relation dialog box, click the **Triggers** tab.

2. Select the trigger you want to remove.

3. Click **Remove**.

Defining Connection Ends

The **From** and **To** areas on the New Relation dialog box define the ends of the relationship connections. You must define the:

- Types of business objects and connections that can have this relationship.

It is important to note that the rules governing the creation of business objects and connections (e.g. definitions of type, relationship type, policy, etc.) are only enforced at the time creation. These rules will be ignored during later modifications of those business objects and connections.

For example, changing the type of business object on one end of a connection will ignore the type restrictions defined on the relationship. This behavior is allowed to support a dynamic modeling environment.

- Meaning of each connection end.
- Rules for maintaining the relationship.

You can define a relationship between two business objects, two connections, or a business object and a connection. When looking at a relationship, the objects are connected "from" and "to" one another.

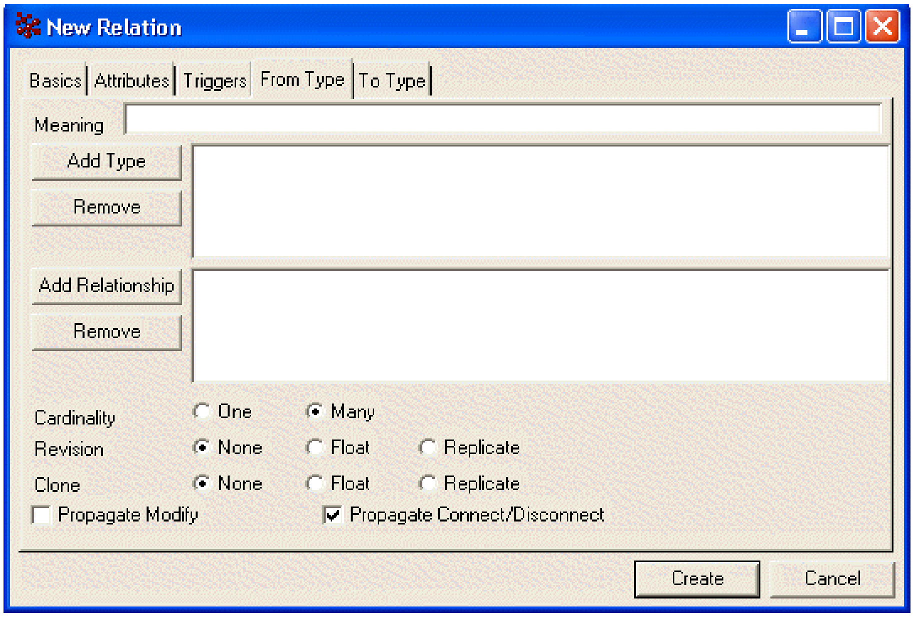
In some relationships, you can assign either object to either end (From or To). However, at times there may be a distinction between the From and To ends. The meaning assigned to the objects on each end of the relationship helps distinguish which object type should be on each end. Regardless of whether an end object is subordinate or equal to the other, you must define both ends of the connection: From and To.

A relationships will also accept the child types of any types assigned to its connection ends. For more information on inherited types, see *Assigning a Parent Type* in Chapter 4.

The procedure for defining both ends is the same. When you define an end of a connection, you specify the following:

- **Meaning**—The Meaning helps identify the purpose of the objects being connected.
- **Type**—The Type defines the types of business objects you can use for that end of the relationship.
- **Relationship**—The Relationship defines the types of connections you can use for that end of the relationship.
- **Cardinality**—Cardinality indicates the number of connections of this type that a single object can have.
- **Revision**—The Revision setting determines how those connections are maintained when one of the connected objects is revised.
- **Clone**—The Clone setting determines how those connections are maintained when one of the connected objects is cloned.
- **Propagate settings**—You can choose whether or not to timestamp the objects on the ends when the relationship's attributes are modified and/or when the relationship is created or deleted (connect or disconnect occurs).

To define connection ends, click the **From Type** or the **To Type** tab in the New Relation dialog box. Both dialogs have the same options. The following shows the **From Type** tab:



Cardinality ○ One ● Many

Revision ● None ○ Float ○ Replicate

Clone ● None ○ Float ○ Replicate

□ Propagate Modify ☑ Propagate Connect/Disconnect

Defining a Meaning

An end’s *meaning* is a descriptive phrase that identifies how the connection end relates to the other end (when viewed from the other end). The meaning helps the user identify the purpose of the objects being connected. Although the meaning is not required, it is strongly recommended. The meaning field is limited to 127 characters.

The meaning is particularly important when there is a distinction between the two ends. It tells the user the difference.

Even when both objects are equivalent, entering a meaning is helpful. It tells the user that the order does not matter. For example, assume you wrote a relationship definition to link equivalent components. The meaning could state that both objects have the same meaning. While the user might guess the meaning from the relationship name, the meaning eliminates doubt.

Adding Type(s) and Relationship(s)

When you define a relationship, you specify the types and/or other relationships that are allowed at each end of the relationship. You must specify at least one business object type or relationship at each end in order for the relationship to be valid.

The types and relationships you specify must already be defined in Live Collaboration.

To select types of business objects

1. Click **Add Type**.

   The Type Chooser opens.

2. Select one or more types and click **OK**. Use Ctrl-click to select multiple types.

To select Relationships between business objects

1. Click Add Relationship.

The Relationship Chooser opens.

2. Select one or more relationship and click OK. Use Ctrl-click to select multiple relationships.

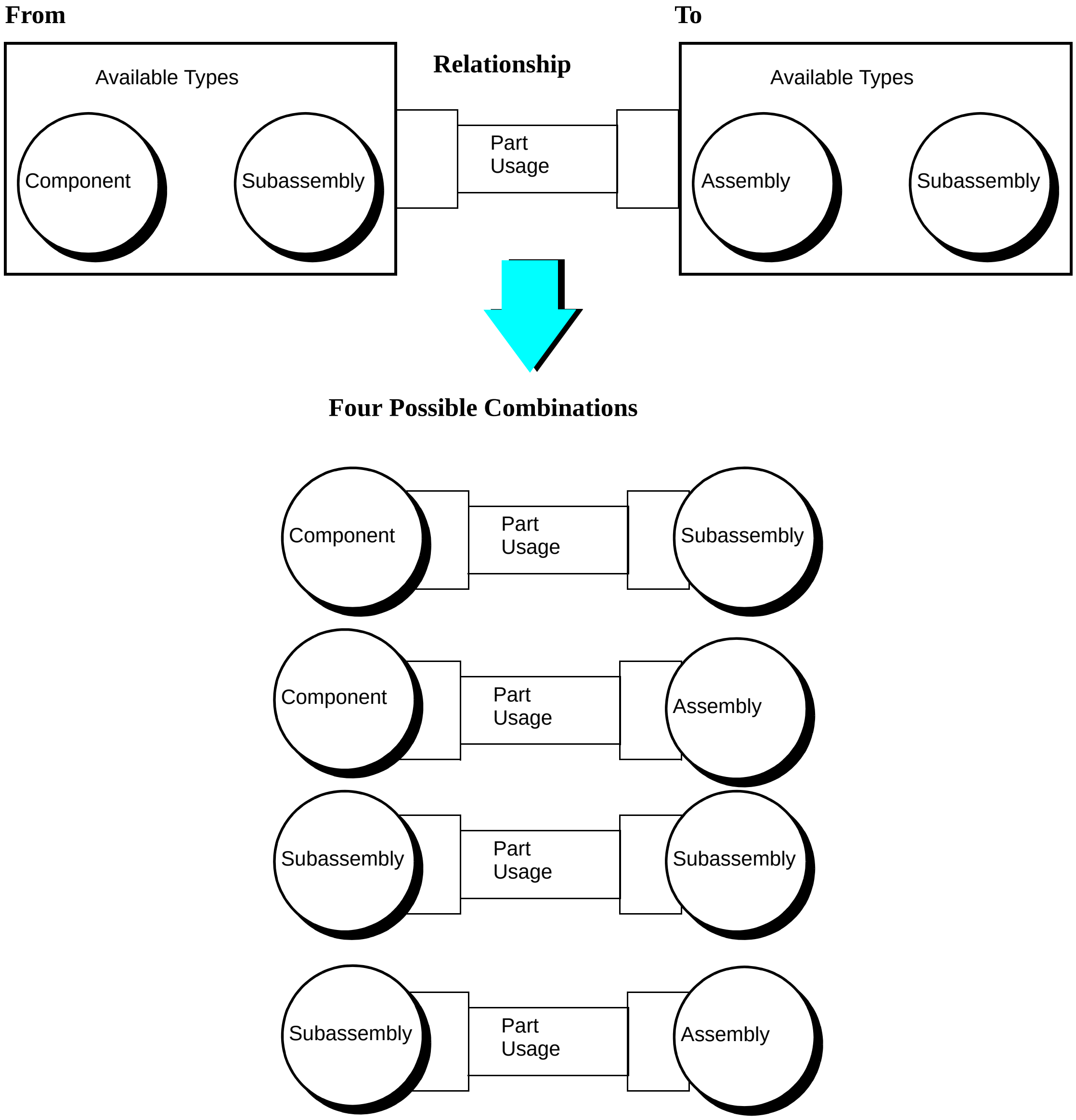
   The selected types and relationships are listed in the From Type or To Type dialog tabs.

When both ends of the connection involve a single business object type, the relationship name can reflect the types being connected. For example, a relationship name of “Drawing and Model” might always refer to a connection between an electronic drawing and a physical model made from the drawing. A name of “Alternative Component” might always refer to a connection between two component objects used interchangeably in an assembly. In both examples, the name reflects the type of objects connected by the relationship.

When a connection end can be assigned multiple business types or relationships, the name of the relationship needs to be more generic. For example, a relationship named “Part Usage” might have one connection end that is either an Assembly or Subassembly object type. The other connection end is a either a Component or Subassembly object type. While you have only one relationship, you actually have four types of connections you can make:

- Component objects and Subassembly objects
- Component objects and Assembly objects
- Subassembly objects and Subassembly objects
- Subassembly objects and Assembly objects

As shown in the following figure, this enables you to relate all components and subassemblies to their larger subassemblies and assemblies without defining a relationship for each connection type.



To remove an assigned type

1. In the New Relation dialog box, click the **From Type** or **To Type** tab.
2. Select the type you want to remove.
3. Click **Remove**.

To remove an assigned relationship

1. In the New Relation dialog box, click the **From Type** or **To Type** tab.
2. Select the relationship you want to remove.
3. Click **Remove**.

In the sections that follow, the use of the term "object" means both business objects and relationships.

Defining Cardinality

Cardinality refers to the number of *connections of this type* that objects can have. When you define the cardinality, it can be either:

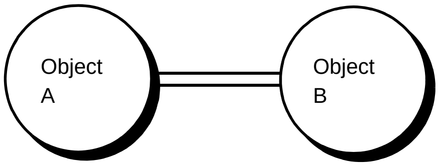
- **One**—The object can only have one connection of this relationship type at any time.

  Or

- **Many**—The object can have several connections of this type simultaneously.

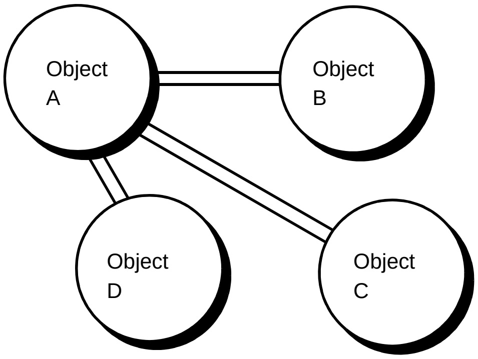
Since cardinality is defined for each end of a connection, there are three possibilities:

ONE-to-ONE



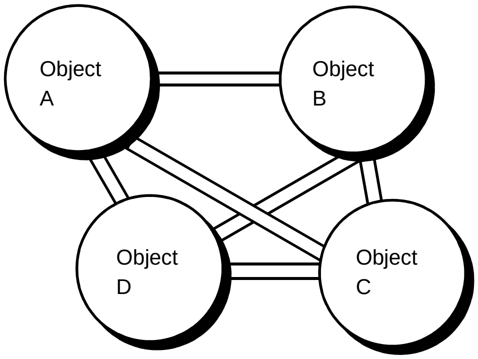
Object A can connect only with Object B.

ONE-to-MANY
or **MANY-to-ONE**



Objects B, C, and D can have only one connection. Object A can have many connections.

MANY-to-MANY



All objects can have multiple connections simultaneously.

Tip: Think of how many objects will typically exist on each end of the relationship. If it is one, the cardinality is One. If it is more than one, the cardinality is Many.

One-to-One

In a One-to-One relationship, the object on each end can be connected to only one other object with this type of relationship. An example of this type of cardinality might apply with a Change Order object connected to a Drawing object. Only one Change Order can be attached to the Drawing at any time and only one Drawing object can be attached to a Change Order.

One-to-Many or Many-to-One

In a One-to-Many or Many-to-One relationship, the object on the “many” side can be attached to many other objects with this relationship type while the object on the “one” end cannot. An example of this type of relationship is a training course with multiple course evaluations. In an evaluation relationship, a single Training Course object can have many Course Evaluation objects attached to it. Therefore, the side of the relationship that allows the Training Course type needs a cardinality of One so that each Course Evaluation object can be connected to only one Training Course. On the other hand, the side of the relationship that allows the Course Evaluation type needs a cardinality of Many to allow many of them to use this kind of relationship to attach to the Course object.

Tip: Think of how many objects will typically exist on each end of the relationship. If it is one, the cardinality is ONE. If it is more than one, the cardinality is MANY.

Many-to-Many

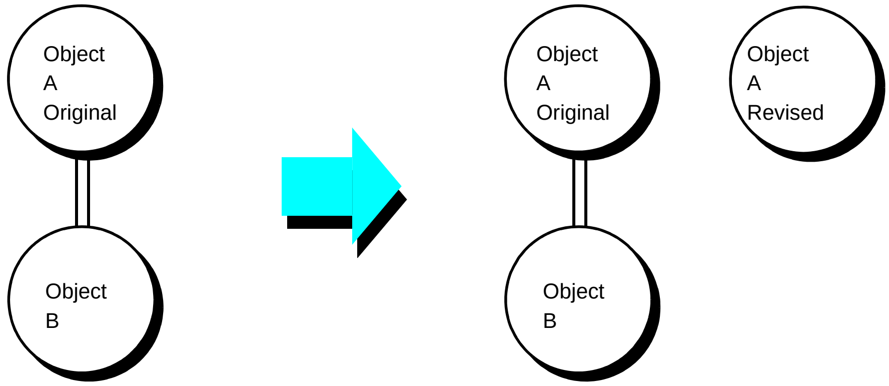
In a Many-to-Many relationship, objects on both ends of the relationship can have multiple simultaneous connections of this relationship type. This type of cardinality is evident in a relationship between Component and Assembly objects. One Component object can be simultaneously connected to many different Assembly objects while one Assembly object can be simultaneously connected to many different Component objects. Both sides of the relationship are defined with a cardinality of Many since both can have more than one connection of this type at any time.

Selecting the Revision Rule

When you are defining the cardinality value for a connection end, one factor that you must consider is revision. What will happen to the relationship if one of the connection ends is revised? Will you shift the relationship to the revised object, create a second new relationship with the revised object, or simply maintain the status quo by retaining a relationship with the unrevised object? The answer is specified by the revision rule associated with each connection end, as described in the following paragraphs.

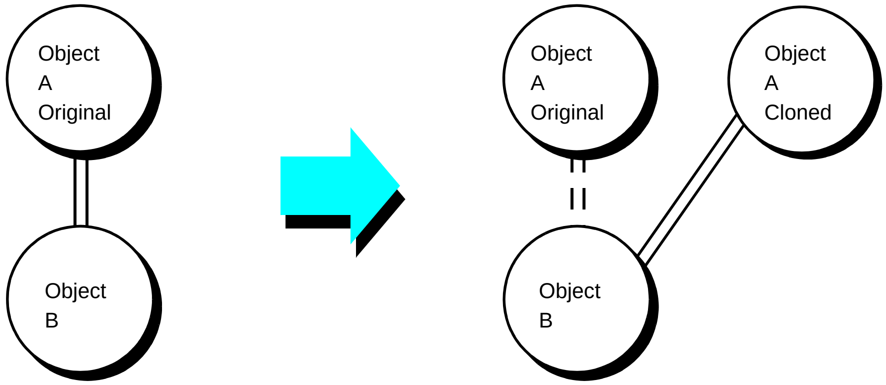
The *revision rule* specifies how revisions of the connected object are handled. These rules are intended only as aids to users and reduce much manual connection activity. They are not designed to automatically maintain configurations (such as in product structures). There are three revision rules: **None**, **Float**, and **Replicate** (as illustrated below).

NONE



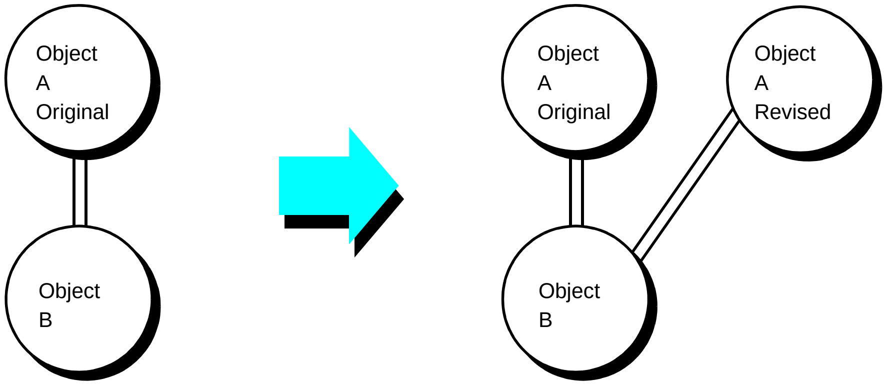
The revised object has no connection.

FLOAT



The connection shifts to the revised object.

REPLICATE



A new connection is made to the revised object. The original connection is left intact.

None

When a connection end uses the None revision rule, nothing happens to the established connection when the end object is revised. This means that the revised object will not automatically have a connection associated with it.

Using the None revision rule is useful when an object revision removes the need for the connection. For example, when you have a connection between a Training Course object and a Course Evaluation object, the connection may no longer be required or useful if the Training Course object is revised. While you may want to maintain the connection between the old version of the Training Course and the evaluation, the evaluation does not apply to the new version of the Training Course object. Therefore the connection end occupied by the Training Course object would use the None revision rule. But what of the other connection end?

If the Evaluation object is revised, it is still useful to the Training Course object. The revised object should remain connected to the Training Course object but the unrevised object no longer applies. To handle this situation, you would define the Course Evaluation end with the Float revision rule.

Float

The Float revision rule specifies that the relationship should be shifted whenever the object is revised. When the Float revision rule is used, the unrevised (or older version of the) object loses the connection with its other end. In its place the other end is automatically connected to the revised object. Now the older version of the object will be unattached while the newer version will have the relationship. This floating of the connection ensures that the latest versions of the object(s) are linked together.

But what if you want to maintain the old relationship while still creating a relationship with the new version? This would actually produce two connections: one between the unrevised object and its other end and one between the revised object and its other end. In this case, you would use the Replicate revision rule.

Replicate

The Replicate revision rule automatically makes new connections whenever an object on that end is revised. This results in an object that may have more than one simultaneous connection of the same relationship type. For this reason, any connection end that uses the Replicate revision rule must also use a cardinality of Many as described earlier in this chapter. If a cardinality value of One is used, Replicate cannot work.

The Replicate revision rule is useful when you want to keep track of former connections. Since the old connections are maintained while new ones are created, a user can easily see all revisions that are related to a connected object. For example, you might have a relationship between a Specification object and a Specification Change object. As the Specification object is revised, you want to maintain the relationship so that you know that this Specification Change applied to this revision of the Specification.

However, you also want the relationship to exist between the revised Specification and the Specification Change. This new relationship enables you to trace the history of the changes made and the reasons for them. in this situation, the Specification object should use the Replicate revision rule while the Specification Change object might use the Float or Replicate revision rule.

Selecting the Clone Rule

Just as you must define what should happen to the relationship if one of the connection ends is revised, you must define what should happen if one of the connection ends is cloned. The same three rules available for revisions are available for clones: None, Float, and Replicate. For a complete description of how each rule works, see *Selecting the Revision Rule* above.

Most business rules require a clone to be treated much differently than a revision, so you may often select a different rule for clones than for revisions. For example, the None rule is often useful when a connection end is cloned. Consider a Training Course object that is cloned because a second course is now being offered by a new instructor. The course evaluations for the original Training Course object do not apply to the second course so the connection to the new Training Course object is not needed.

Selecting Propagate Modify

The **Propagate Modify** switch controls whether or not changes to a connection affect the modification timestamp of the “From” and/or “To” object. If activated, the modified time/date stamp of the appropriate object(s) is updated whenever a connection’s attributes are modified. This allows queries that find all objects that have been updated on a specified date to include objects whose relationships have been modified.

Selecting Propagate Connect/Disconnect

The **Propagate Connect/Disconnect** switch controls whether or not connections and disconnections affect the modification timestamp of the “From” and/or “To” object. If activated, the modified time/date stamp of the appropriate object(s) is updated whenever a connection is created or deleted. This allows queries that find all objects that have been updated on a specified date to include objects that have been connected or disconnected from other objects.

Creating the Relationship

After you enter all appropriate information on the New Relation dialog box, click **Create**. A system message tells you the relationship is being created.

If you see an error message, correct the definition as required and click **Create** again.

6

Working With Interfaces

Interfaces Defined

You may have the need to organize data under more than one classification type. For instance, a Part may have a classification based on its function, which is most typical, but it may also require classification on other issues such as production process, manufacturing location, etc. For each classification type, there is typically a collection of attributes that can be defined for each instance of the classification type and used for searching.

An *Interface* is a group of attributes that can be added to business objects as well as connections to provide additional classification capabilities in Live Collaboration. When an Interface is created, it is linked to (previously-defined) attributes that logically go together. Interfaces are defined by the Business Administrator and are added to business objects or relationship instances with MQL.

An Interface can be *derived* from other Interfaces, similar to how types can be derived. Derived Interfaces include the attributes of their parents, as well as any other attributes associated with it directly. The types associated with the parents are also associated with the child.

The primary reason to add an interface to a business object or a connection is to add the attributes to the instance that were not defined on the type. Moreover, when you add an interface to a business object or a connection, it gives you the ability to classify it by virtue of the interface hierarchy.

Attribute values that come from interfaces cannot be used in a create access rule, because the interfaces are not applied until AFTER the create access checks.

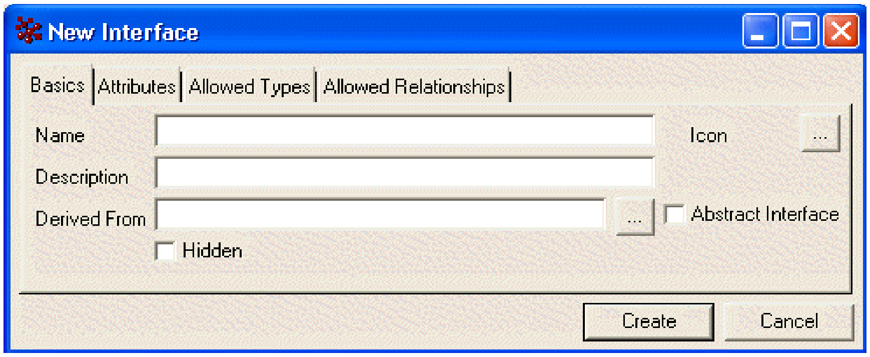
Defining an interface

There are several parameters that can be associated with an interface. Each parameter enables you to provide information about the new interface. While only a name is required, the other parameter values can further provide useful information about the interface.

To define an interface

1. Click or select **Interface** from the Object>New menu.

   The New Interface dialog box opens, showing the **Basics** tab.



2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

You can specify the name of the interface you are creating. All interfaces must have a unique name assigned. The interface name is limited to 127 characters.

Consider the following examples:

`Metal`
`Cars`
`Gold`

The interface name will appear whenever the interface is listed in a Live Collaboration window.

You can define an interface by simply assigning a name to it. If you do, it is assumed that the interface is non-abstract, uses the default interface icon, does not contain any explicit attributes, and does not inherit from any other interfaces.

Defining the Description

A description provides general information about the function of the interface you are defining. There may be subtle differences between interfaces; the Description enables you to point out the differences in the interfaces.

There is no limit to the number of characters you can include in the description. The description can be a prompt, a comment, or a qualifying phrase. For example, if you are defining an interface named Drawing, you might enter this description:

```
Material used to Manufacture
```

Assigning an Icon

You can assign a special icon to the new interface or use the default icon. Icons help users locate and recognize items. The default icon is used when in view-by-icon mode. Any special icon you assign is used when in view-by-image mode. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

GIF filenames should not include the @ sign, as that is used internally by ENOVIA Live Collaboration.

Assigning a Parent Interface

Interfaces can inherit from other interfaces and can be abstract and non-abstract.

- *Abstract interfaces* act as categories for other interfaces. They are not used to add attributes to a business object directly and are useful only in defining characteristics that are inherited by other interfaces. For example, you could create an abstract interface called “Metal.” Two non-abstract interfaces, “Aluminum” and “Iron”, could inherit from it. See *Defining an Abstract Interface* for more information.
- *Non-abstract interfaces* are used to add groups of attributes to a business object. You can add non-abstract interfaces to business objects. For example, assume that Aluminum is a non-abstract interface with its own set of attributes. You can add this interface to objects that are manufactured from aluminum and therefore need this set of attributes.

Use the Derived From box to identify 1 or more existing interfaces as the parent(s) of the interface you are creating. The parent interface(s) can be abstract or non-abstract. A child interface inherits the following items from the parent(s):

- all attributes
- all types to which it can be added
- all relationships to which it can be added

Assigning a parent interface is an efficient way to define several object interfaces that are similar because you only have to define the common items for one interface, the parent, instead of defining them for each interface. The child interface inherits all the items listed above from the parent but you can also (and probably will) add attributes directly to the child interface. Any changes you make for the parent are also applied to the child interface.

For example, suppose you have an interface named “Manufacturing Process”, which includes 2 attributes: “Process” and “Vendor”. Now you create two new interfaces named “Rolled” and “Molded” and specify that they are derived from the interface “Manufacturing Process”.

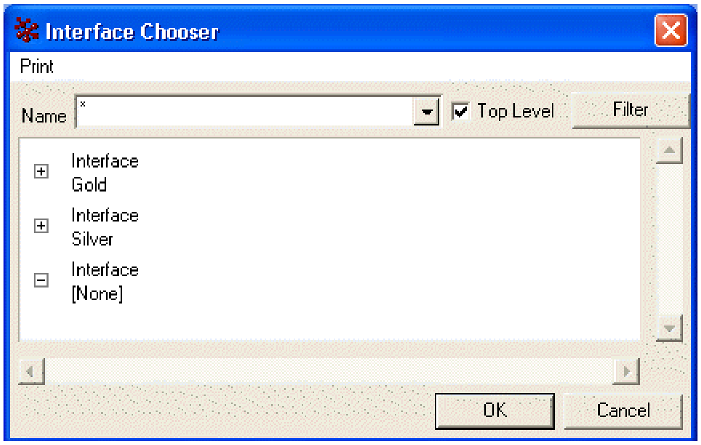
Both new interfaces acquire the attributes in the Manufacturing Process interface. Rather than adding each of these attributes to the new interfaces, you can make Manufacturing Process the parent of the new interfaces. The new interfaces then inherit the 2 attributes as well as the allowed Types and Relationships from the parent.

Interfaces can have more than 1 parent.

To assign a parent interface

- Type the interface name in the **Derived From** text box.

  Or
- Click the **Derived From** ellipsis button and select a interface from the Interface Chooser that appears.



All attributes defined for the parent interface are added to the definition for the child. These items include the word “Inherited” so you can distinguish them from attributes you assigned directly.

Defining an Abstract Interface

Abstract interfaces act as categories for other interfaces.

To define an abstract interface

- Check the **Abstract Interface** checkbox.

  Interfaces are not abstract by default.

Abstract interfaces are helpful because you do not have to reenter attributes that are often reused. If an additional field is required, it needs to be added only once. For example, the “Person Record” object interface might include a person’s name, telephone number, home address, and social security number. While it is a commonly used set of attributes, it is unlikely that this information would appear on its own. Therefore, you might want to define this object interface as an abstract interface.

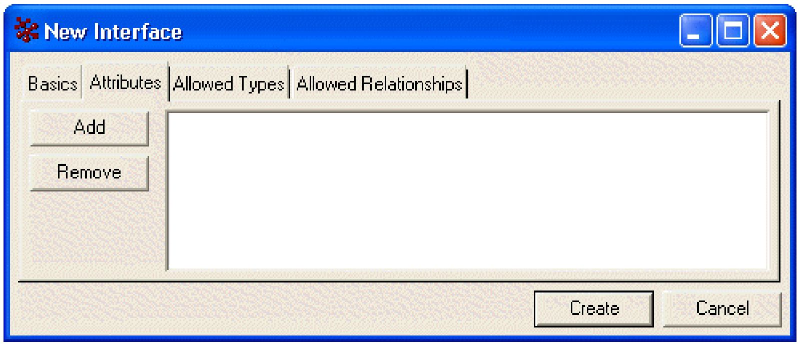
Defining the Hidden Option

You can specify that the new interface is “hidden” so that it does not appear in any dialogs that list interfaces. You may want to use the hidden option if, for example, an object is under development or if it is intended only for your personal use. Users who are aware of the hidden interface’s existence can enter its name manually where appropriate.

Hidden objects are always accessible through MQL.

Selecting Attributes

An interface can have any combination of attributes associated with it. To select attributes, click the **Attributes** tab in the New Interface dialog box. The following dialog box opens:



To assign attributes to the interface

1. Click **Add**.

   The Attribute Chooser opens.

2. Select one or more attributes in the dialog box. Use Ctrl-click to select multiple attributes.

3. Click **OK**.

   The selected attributes are listed in the **Attributes** dialog box.

If you select an attribute that is a duplicate of an attribute assigned through inheritance, the duplication is ignored. Live Collaboration recognizes that the attributes are the same.

If you add an attribute that is part of an index, the index is disabled. Refer the MQL Guide Index command for more information.

To remove an assigned attribute

1. In the New Interface dialog box, click the **Attributes** tab.

2. Select the attribute you want to remove.

3. Click **Remove**.

You can view or edit an assigned attribute by double-clicking the attribute icon. For information on editing an attribute, see Defining an Attribute in Chapter 2.

Adding an attribute to an interface should not be included in a transaction with other extensive operations, especially against a distributed database. This is a “special” administrative edit, in that it needs to update all business objects that use the interface with a default attribute.

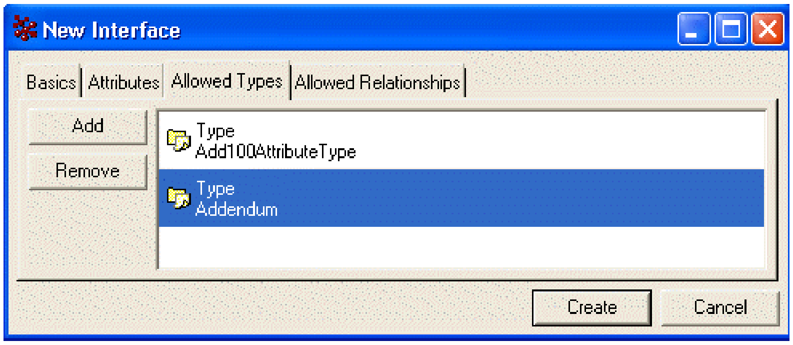
For the Matrix Navigator user, when viewing attributes, they will appear in the reverse order of the order in which you chose them. Therefore, you select the attribute you want first last.

Selecting Types

You can define all the business object types that can use the interface. An interface may be allowed for use with any number of types, and likewise, a type may be allowed to use any number of interfaces.

Choose a type or a relationship to make the interface usable.

To select types, click the **Allowed Types** tab in the New Interface dialog box. The following dialog box opens:



Type
Add100AttributeType

Type
Addendum

To assign types to the interface

1. Click **Add**.

   The Type Chooser opens.

2. Select one or more types in the dialog box. Use Ctrl-click to select multiple attributes.

3. Click **OK**.

   The selected types are listed in the **Allowed Types** dialog box.

To remove an assigned type

1. In the New Interface dialog box, click the **Allowed Type** tab.

2. Select the type you want to remove. Use Ctrl-click to select multiple types.

3. Click **Remove**.

You can view or edit an assigned type by double-clicking the type icon. For information on editing a type, see Working With Types in Chapter 4.

Selecting Relationship

You can define all the relationships that can use the interface. To select relationships, click the **Allowed Relationships** tab in the New Interface dialog box.

To assign relationships to the interface

1. Click **Add**.

   The Relationship Chooser opens.

2. Select one or more relationships in the dialog box. Use Ctrl-click to select multiple relationships.

3. Click **OK**.

The selected relationships are listed in the **Allowed Relationships** dialog box.

To remove an assigned relationship

1. In the New Interface dialog box, click the **Allowed Relationships** tab.
2. Select the relationship you want to remove. Use Ctrl-click to select multiple relationships.
3. Click **Remove**.

You can view or edit an assigned relationship by double-clicking the relationship icon. For information on editing a relationship, see Working With Relationships in Chapter 5.

Creating the Interface

After you enter all appropriate information in the New Interface dialog box, click **Create**. If you see an error message, correct the interface definition as required and click **Create** again.

Copying and Modifying an Interface

Copying (Cloning) an Interface Definition

After an interface is defined, you can clone the definition. Refer to the description in *Cloning an Administrative Object*.

Modifying an Interface Definition

After an interface is defined, you can change the definition with the Edit Interface dialog. Refer to the description in *Modifying an Administrative Object*.

You can make the following modifications:

- change the name
- change the description
- assign a different icon for the interface
- assign a different parent
- Specify the interface to be abstract or hidden
- add or remove attributes
- add or remove allowable types
- add or remove allowable relationships

Adding and removing attributes to an interface have the same effect as adding and removing them from a Type or Relationship. If many objects use the interface, many database rows are affected by the change, and so the transaction has the potential to be very time-consuming.

7

Working With Formats

Overview

A *format* definition is used to capture information about different application file formats. A format stores the name of the application, the product version, and the suffix (extension) used to identify files. It also contains the commands necessary to launch the application automatically and load the relevant files from Live Collaboration. Formats are the definitions used to link Live Collaboration to the other applications in the users' environment.

Applications typically change their internal file format from time to time. Eventually older file formats are no longer readable by the current version of the software. It is wise to create new format definitions (with appropriate names) as the applications change so that you can later find the files that are in the old format and bring them up to date.

A business object can have many file formats and they are linked to the appropriate type definition by the policy definition (see *Selecting a Format in Chapter 9*).

Defining a Format

There are several parameters that can be associated with a format. Each parameter enables you to provide information about the new format.

Using MQL, you can specify a MIME type *for a format.* MIME types are *used when files are accessed via a Web browser. For information, see the MQL Guide : Relationships.*

To define a format

1. Click 📄 or select **Format** from the Object>New menu.

   The New Format dialog box opens.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

You can specify the name of the format you are creating. All formats must have a unique name. You should assign a name that has meaning to both you and the user. The format name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

For example, assume there are two types of word processing programs commonly used to create text documents, BestWord and Writer's Aid. You can create two formats, one for each word processing program:

```
BestWord
Writer's Aid
```

Each format will define the word processing program that will be used to view, edit, and print a file created with that program. The format name appears whenever the format is listed in a window.

The Default File Suffix specified in the Format is not used in the launching mechanism—the file itself is passed to the operating system and its extension (or suffix) is used to determine what application should be opened.

Assigning an Icon

You can assign a special icon to the new format. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Version

A version identifies the version number of the software required to process the file. The version of the software is useful when tracking files that were made under different software releases. Upward and downward compatibility is not always assured between releases. If you install a new software release that changes the internal format of the application's files, you can create a new format specifying the new version.

For example, the following are valid values for the version:

```
3.1
```

```
A
Standard
```

The version is displayed with the format name whenever it appears in a window.

Defining a Description

A description provides general information to both you and other Business Administrators about the types of files associated with this format and the overall function of the format.

For example, assume you want to write descriptions for two word processing formats. For the format named “BestWord” you might assign a description such as:

```
Use for memos, letters, and other correspondence
created with BestWord software
```

For the format named “Writer’s Aid” you might write a description such as:

```
Use for technical documents (user guides, etc.)
created with Writer’s Aid software.
```

These descriptions help clarify the purpose of the format.

There is no limit to the number of characters you can include in the description. However, keep in mind that the description appears when the mouse pointer stops over the format in a chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Defining a Default File Suffix

You can specify the default suffix of the file. This is used for all file names created by Live Collaboration using this format. Occasionally Live Collaboration is required to create and name a file within a business object. For example, Live Collaboration will automatically create a file when a user creates an annotation or attachment in Live Collaboration. When the file is created, it has a name that contains two parts: the name of the object and a suffix. The suffix appears only if you assigned it to the associated format prior to the creation of the file.

For example, assume you want to add a note to a business object. You might use either the BestWord or Writer’s Aid word processing software to create the note. BestWord uses a default file suffix of “.bw” for files and Writer’s Aid uses the “.wa” suffix. These suffixes allow you and others to quickly distinguish the type of file.

Defining a Creator and Type

The Creator and Type fields are Macintosh file system attributes (like Protection and Owner on UNIX systems). They should not be confused with Live Collaboration users or types. An example is:

```
creator=’MPSX’, type=’TEXT’
```

This would identify a script file created by the Macintosh toolserver. Both fields are four bytes in length and are generally readable ASCII. The values for creator and type are registered with Apple for each Macintosh application. When a file is checked out to a Macintosh, these attribute settings will be applied. If Macintoshes are not used, the fields can be left blank.

Defining a View, Edit, and/or Print Command

The View, Edit, and Print Command fields specify the program to use to view (open for view), edit (open for edit), or print files checked into the format. When you specify the program, you are actually specifying the name of the program object that represents the program.

For Windows platforms, if you want to open files for view, edit, or print based on their file extensions and definitions in the Windows Registry, you can leave the corresponding box blank. For example, by default Windows uses MS Paint to open files with a file extension of .bmp. Keep in mind that each user's PC contains its own Windows Registry database, which is editable; the databases are not shared between computers.

Program object requirements

To be used in a format definition, a program object definition must include these characteristics:

- The Needs Business Object box must be checked.
- The Code tab must contain the command needed to execute the program and the syntax for the command must be appropriate for the operating system.
- The Code tab content should end with the $FILENAME macro so the program opens any file. Enclose the macro in quotes to ensure that files with spaces in their names are opened correctly.

Defining the commands

To define a view, edit, or print command

1. Type the name of the program to run. The program must be a program object in Live Collaboration and it must be defined as a program that needs a business object.

   Or

   Click 🖥 to display the Program Chooser. Only programs that need business objects are listed. Select the program to run and click **OK**.

2. Ensure the program opens files with spaces in their names by editing the program object so it includes the $FILENAME macro:

   a) Click 📝 to display the Edit Program dialog box.

   b) In the code tab, type "${FILENAME}" after the program path and make sure you include the quotation marks.

Defining the Hidden Option

You can mark the new format as "hidden" so that it does not appear in the checkin/ checkout list of formats. You may want to use the hidden option if, for example, an object is under development or if it is intended only for your personal use. Hidden objects are accessible through MQL.

Creating the Format

After you enter all appropriate information on the New Format dialog box, click **Create**. A system message tells you the format is being created.

Modifying a Format Definition

After you establish a format definition, you can add or remove defining values. Refer to the description of *Modifying an Administrative Object* in Chapter 1.

When modifying a format, consider upward and downward compatibility between software versions. For example, files that can be processed with an earlier version may not be usable with a later version. Since all files and the defined format will be affected if you modify the format definition, you should test sample files or read the software release notes to determine if old files will be affected negatively. If they will be affected, you may want to create a new format for the new software version rather than modify the existing format definition.

8

Working With Rules

Overview

Use rules to limit user access to specific attributes, forms, programs, and relationships. Unlike policies, rules control access to these administrative objects regardless of the object type or state. For example, a policy might allow all users in the Engineering group to modify the properties of Design Specification objects when the objects are in the Planning state. But you could create a rule to prevent those users from changing a particular attribute of Design Specifications, such as the Due Date. In such a case, Engineering group users would be unable to modify the Due Date attribute, no matter what type of object the attribute is attached to.

When you create a rule, you define access using the three general categories used for assigning access in policies: public, owner, and user (specific person, group, role, or association).

IMPORTANT: When adding or modifying accesses, use MQL. There are settings available in MQL that are not available in Business Modeler, such as those used to configure modern access security used by apps.

Creating Rules

To define a rule

1. Click or select **Rule** from the Object>New menu.

   The New Rule dialog box opens.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

All rules must have a unique name assigned. You should assign a name that has meaning to both you and other business administrators. The field limit is 127 characters. Refer to *Administrative Object Names* in Chapter 1 for a list of characters you should avoid using.

You could have two rules that control access for the Marketing group. These rules might be named Marketing and Marketing Managers. Marketing Managers may be allowed extra privileges that the rest of Marketing is not.

Defining a Description

You can provide general information to you and other Business Administrators about what the rule governs and the accesses assigned in the rule. When you assign a rule to an administrative object, the description could inform you of who has special access. Since there may be subtle differences between rules, the description can point out these differences.

There is no limit to the number of characters you can include in the description. However, keep in mind that the description is displayed when the mouse pointer stops over the rule in the chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Defining the Hidden Option

You can mark the new rule as “hidden” so it does not appear in the rule chooser which simplifies the end-user interface. Business Administrators who are aware of the hidden rule’s existence can enter its name manually where appropriate.

Defining User Access

The **Access** tab lets you specify who should have access to the governed administrative objects and how much access they should have. For every administrative object (attribute, form, program, relationship) governed by a rule, a user has access only if the rule specifically grants the user access. If the user isn’t granted access by the rule, the user won’t have access, even if the policy grants access to the user.

IMPORTANT: When adding or modifying accesses, use MQL. There are settings available in MQL that are not available in Business Modeler, such as those used to configure modern access security used by apps.

For example, suppose you don’t want anyone to modify an attribute called Priority unless they belong to the Management role. However, everyone should be able to view the attribute’s values. You would create a rule that governs the Priority attribute and define the following accesses:

Owner: read
Public: read
Management role: all

Like policy states, there are three categories of access for rules: **Public**, **Owner**, and **User**. Public refers to everyone in the database, owner refers to the person who creates the business object or to whom the object was routed or reassigned, and user can be defined as any person, group, role, or association. In the **Access** area, you edit an access item to define the access. Keep in mind that the system doesn’t check rules unless the user has already been assigned access in the policy and person definitions.

It is important to keep in mind that while the complete list of access items is available when creating rules, when they are actually used, only the applicable privileges are checked. The table below shows the accesses each administrative type uses:

Accesses Used By:				
Attributes	**Forms**	**Programs**	**Relationships**	
read	viewform	execute	toconnect fromconnect	freeze
modify	modifyform		todisconnect fromdisconnect	thaw
			changetype	modify (attributes)
Owner Access does not apply to Relationships.				

Using a Filter Expression

User access lists defined on a rule can accept a filter expression in order to grant or deny access to a specific user. If the filter expression evaluates to “true,” the specified access will be granted; otherwise the access is denied. This provides an additional level of access granularity, which increases security and reduces overall management problems by reducing the number of policies or rules required.

Expression access filters can be any expression that is valid in the Live Collaboration system. Expressions are supported in various modules of the Live Collaboration system, for example, query where clauses, expand, filters defined on workspace objects such as cues, tips and filters, etc.

Since you must have at least read access to the business object in order to evaluate the filter, normal access checks must be disabled while an access filter is being evaluated.

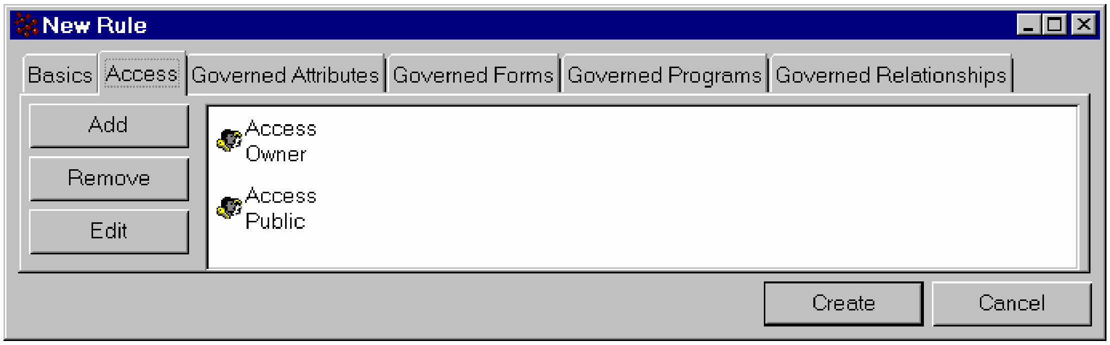
Policies and rules should use organization, project, maturity, and reserve logic instead of access filters whenever possible. These security checks ensure the completeness of search results. You can also use the Configure My ENOVIA app for defining P&O.

To define user accesses for the rule

1. Click the **Access** tab in the New Rule dialog box.

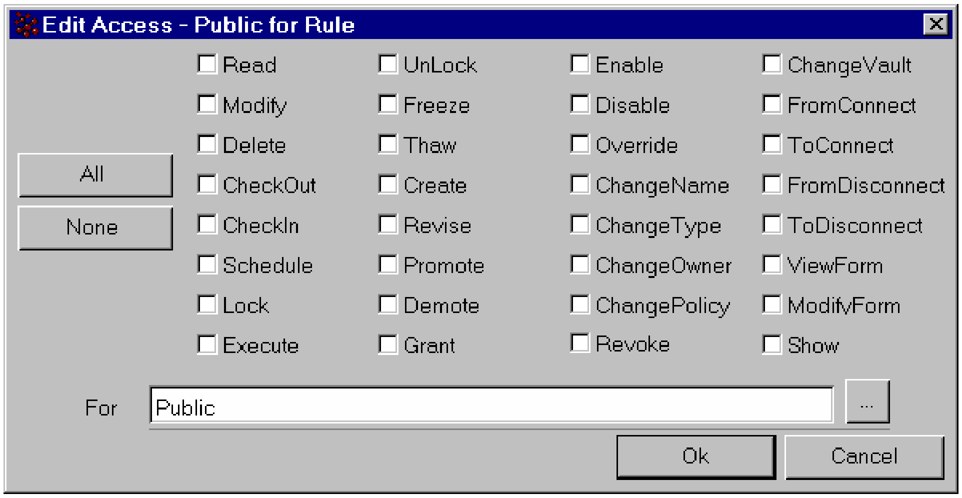
The Owner and Public access categories are already listed. A rule always has one Public access and one Owner access item. However, User access items can be quite numerous. For example, you might want to specify different levels of access for different user categories—persons, groups, roles, and associations.

For a description of all the accesses, see the *Administration Guide : Accesses*.



2. Define the accesses for the public.

a) Select the **Public Access** item and click **Edit**.



□ Read
□ Modify
□ Delete
□ CheckOut
□ CheckIn
□ Schedule
□ Lock
□ Execute
□ UnLock
□ Freeze
□ Thaw
□ Create
□ Revise
□ Promote
□ Demote
□ Grant
□ Enable
□ Disable
□ Override
□ ChangeName
□ ChangeType
□ ChangeOwner
□ ChangePolicy
□ Revoke
□ ChangeVault
□ FromConnect
□ ToConnect
□ FromDisconnect
□ ToDisconnect
□ ViewForm
□ ModifyForm
□ Show

For: Public ...

b) In the Edit Access dialog box, check the accesses you want all users in the database to have for the rule. The public should typically have few or no accesses. You should only consider accesses that apply to the administrative object governed by the rule. For example, if the rule governs only programs, the only access that applies is Execute.

c) Click **OK**.

3. Define the accesses for the owner.

Owner Access does not apply when the rule governs relationships because relationships don't have owners.

a) Select the **Owner Access** item and click **Edit**.

b) Check the accesses you want the object owner to have for the rule. Owners typically have full access.

c) Click **OK**.

4. Define user accesses.

a) From the Access tab in the New Rule dialog, click **Add**.

A User item is added.

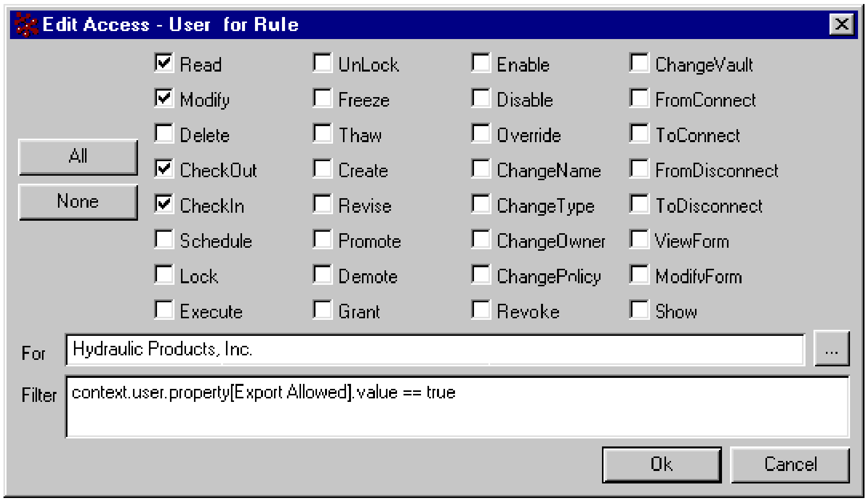
b) Select the **User Access** item and click **Edit**.

c) Click the **Browse ...** button on the right side of the Edit Access dialog box.

d) In the User Chooser, select the person, group, role, or association for which you want to assign access.

e) Check the accesses you want to assign to the user.

f) Optionally, add a filter in the **Filter** text box. (See *Using a Filter Expression* for details.) When the expression evaluates to "true," the specified access is granted. The following is a example of how the user access could be set up in a rule:



Edit Access - User for Rule

☑ Read
☑ Modify
□ Delete
☑ CheckOut
☑ CheckIn
□ Schedule
□ Lock
□ Execute

□ UnLock
□ Freeze
□ Thaw
□ Create
□ Revise
□ Promote
□ Demote
□ Grant

□ Enable
□ Disable
□ Override
□ ChangeName
□ ChangeType
□ ChangeOwner
□ ChangePolicy
□ Revoke

□ ChangeVault
□ FromConnect
□ ToConnect
□ FromDisconnect
□ ToDisconnect
□ ViewForm
□ ModifyForm
□ Show

For: Hydraulic Products, Inc.

Filter: context.user.property[Export Allowed].value == true

In this case, the selected access (read, modify, checkout, checkin) would be allowed only:

- - if context user belongs to the group "Hydraulic Products, Inc." and
- - if `context.user.property[Export Allowed].value` is evaluated as true.

g) Click **OK**.

To remove a user access item

1. Select the User item in the **Access** area.

2. Click **Remove**.

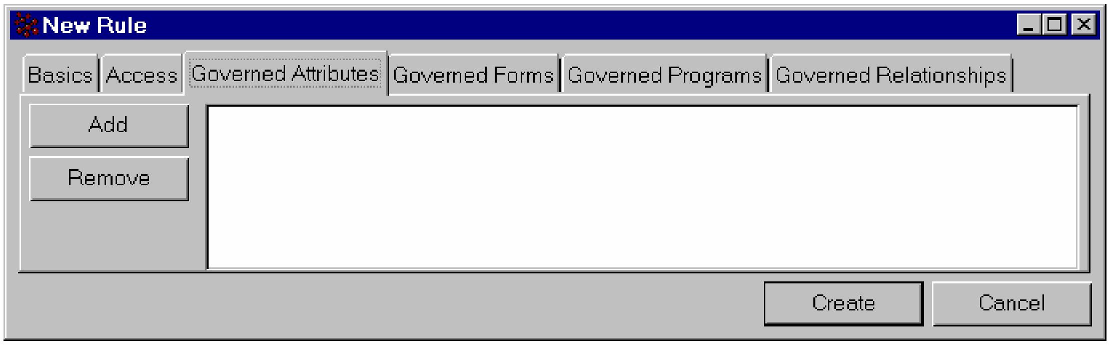
You cannot add or remove the Owner or Public access items.

Assigning the Rule

The "Governed..." tabs allow you to select the attributes, forms, programs, and relationships that the new rule governs. By default, all existing definitions have no rules, and are available to all users. Use the procedure below if you want to add a rule to an existing definition.

To assign the rule to the definitions that will use it

1. The New Rule dialog box has several Governed "item" tabs — **Governed Attributes**, **Governed Forms**, **Governed Programs** and **Governed Relationships**. Click the appropriate tab for your use. Each looks similar to the Governed Attributes tab that is shown below:



2. Click **Add** to display the chooser for that Governed item.
3. Select one or more definitions from the list and click **OK**. The selected definitions appear in the Governed item tab of the New Rule dialog box.

Completing a Rule Definition

After you have entered all of the appropriate information on the New Rule dialog box, click **Create** to create the Rule.

Deleting Rules

If a Rule is no longer required, you can delete it. Refer to the section *Deleting an Administrative Object* in Chapter 1. However, Live Collaboration will not allow you to delete a rule if any object is governed by it. To delete a rule used by any object, you must first reassign the object to another rule or delete the object.

9

Working With Policies

Introduction

A *policy* controls a business object. It governs access, approvals, lifecycle, revisioning, and more. If there is any question as to what you can do with a business object, it is most likely answered by looking at the object's policy.

This chapter begins with extensive discussions about policies. The procedures for defining a policy are in the section *Defining an Object Policy*.

IMPORTANT: When adding or modifying accesses, use MQL. There are settings available in MQL that are not available in Business Modeler, such as those used to configure modern access security used by apps.

Two Sections of a Policy: General Behavior and Lifecycle

A policy is composed of two major sections: one describes the general behavior of the governed objects and the other describes the lifecycle of the objects.

General Behavior

The first section provides general information about the policy. This information includes:

- The types of objects the policy will govern.
- The types of formats that are allowed.
- The default format.
- Where and how checked in files are managed.
- How revisions will be labeled.

Lifecycle

The second section provides information about the lifecycle of the objects governed by the policy. A lifecycle consists of a series of connected states, each of which represents a stage in the life of the governed objects. Depending on the type of object involved, the lifecycle might contain only one state or many states. Branches can be defined to determine what the next state of the object will be based on which signatures requirements are fulfilled. Filters can also be defined to determine which signature requirements are fulfilled. Each state of the lifecycle defines:

- The current state of the object.
- Who will have access to the object.
- The type of access allowed.
- Whether or not the object can be revised.
- Whether or not files within the object can be revised.
- The conditions required for changing state.

Determining Policy States

When creating a policy, defining the policy states is most often the most difficult part. How many states does the policy need? Who should have access to the object at each state and what access should each person have at each state? Should you allow revisions at this state? Should you allow files to be edited? What signatures are required to move the object from one state to another? Can someone override another's signature? As described below, all of these questions should be answered in order to write the state definition section of a policy.

How Many States are Required?

A policy can have only one state or many. For example, you might have a policy that governs photographic images. These images may be of several types and formats, but they do not change their state. In general, they do not undergo dramatic changes or have stages where some people should access them and some should not. In this situation, you might have only one state where access is defined.

Let's examine a situation where you might have several states. Assume you have a policy to govern objects during construction of a house. These objects could have several states such as:

State	Description
Initial Preparation	The building site is evaluated and prepared by the site excavator and builder. After the site is reviewed and all preparations are completed, the excavator and builder sign off on it and the site enters the second state, Framing.
Framing	Carpenters complete the framing of the house and it is evaluated by the builder, architect, and customer. In this state, you may want to prohibit object editing so that only viewing is allowed. If the framing is complete to the satisfaction of the builder, architect, and customer, it is promoted to the third state, Wiring.
Wiring	The electrician wires the house. However, the electrician may sign off on the job as completed only to have the builder reject it. When approval is rejected, promotion to the next state is prevented from taking place.

As the house progresses through the building states, different persons would be involved in deciding whether or not the object is ready for the next state.

When determining how many states an object should have, you need to know:

- What are the states in an object's life.
- Who requires access to the object.
- What type of access they need.

Once a policy is defined, you can alter it even after business objects are created that are governed by it.

Who Will Have Object Access?

There are three general categories used to define who will have access to the object in each state:

- Public—refers to everyone in the Live Collaboration database. When the public has access in a state, any defined Live Collaboration user can work with the business object when it is in that state.
- Owner—refers to the specific person who is the current owner of the object instance. When an object is initially created in the database, the person who created it is identified by Live Collaboration as the *owner* of the object. This person remains the owner unless ownership is transferred to someone else.
- User—refers to a specific person, group, role, or association who will have access to the object. You can include or exclude selected groupings or individuals when defining who will have access.

Is the Object Revisionable?

In each state definition are the terms *Versionable* and *Revisionable*. The term Revisionable indicates whether a new Revision of the object can be made. Versionable is not used at this time, and setting it has no affect on policy behavior.

You can decide when in the object's lifecycle revisions are allowed by setting the switch ON or OFF in each state definition. This setting is independent of who (which person, role or group) has access to perform the operations.

How Do You Change From One State to the Next?

Most often a change in state is controlled by one or more persons, perhaps in a particular role or group. For example, during the construction of a house, the customer and the builder might control the change in state. If you break the building stage down into smaller states, you might have the object's transition controlled by the site excavator, foundation expert, electrician, or plumber. As the house progresses through the building states, different persons would be involved in deciding whether the object is ready for the next state. You certainly would not want the carpenters to begin working before the foundation is done.

In Live Collaboration, signatures are a way to control the change of an object's state. Signatures can be associated with a person, group, role, or association. Most often, they are role-related. When a signature is required, a person must approve the object in order for the object to move on to the next state. If that person does not approve it, the object remains in the current state until the person does approve or until someone with higher authority provides approval.

More than one signature can be associated with the transition of an object. Lifecycles can be set up such that the signature that is approved determines which state is the next in the object's life.

A signature can be approved or rejected. For example, an electrician could say a job is done only to have the builder reject it. When approval is rejected, promotion to the next state is prevented from taking place.

Filters can be defined on a signature requirement to determine if it is fulfilled. If the filter evaluates to true, then the signature requirement is fulfilled. This is useful for adding required signatures that are conditional, dependent on some characteristic of a business object.

In the sections that follow, you will learn more about the actual procedures to define a policy and the object states as well as the procedures that manipulate and display policy definitions.

Defining an Object Policy

There are several parameters that can be associated with a policy. Each parameter enables you to provide information, such as the types of objects that will be governed by the policy, the types of formats that are permitted by the policy, the labeling sequence for revisions, where files governed by the policy are stored, and the states and conditions that make up an object’s lifecycle.

To define a policy

1. Click or select **Policy** from the Object>New menu.

   The New Policy dialog box opens, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

You can specify the name of the policy you are creating. All policies must have a unique policy name assigned. You should assign a name that has meaning to both you and the user. The policy name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

For example, you might have two policies that control the development of software programs for new and existing product lines. These policies might be named “Old Dev. Process” and “New Dev. Process.” While these are descriptive names, they are somewhat ambiguous. Does the “Old Software Dev. Process” identify a previously used process or a process for handling existing products? In this case, you could assign more descriptive names such as “Existing Product Dev.” and “New Product Dev.”

Assigning an Icon

You can assign a special icon to the new policy. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

You can provide general information to you and other Business Administrators about the main features of the policy. When you create a business object that has this policy, the description informs you of what can be done with the object. Since there may be subtle differences between policies, the description can point out these differences.

For example, assume you have two engineering development processes. One process is used by the Software Development group and the other is used by the Hardware Development group. These two processes might share some of the same states, types, formats and roles. However, the processes have different people involved as well as different requirements for development and release. You could document the differences in the description. For the policy named “New Product Dev.” you might enter:

```
This process is for developing new standalone products
```

For the policy named “Existing Product Dev.” you might enter:

```
This process is for developing enhancements to existing
software.
```

There is no limit to the number of characters you can include in the description. However, keep in mind that the description appears when the mouse pointer stops over the policy in a chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Defining a Revision Sequence

A sequence defines a scheme for labeling revisions. You can specify the pattern to use when an existing object is revised. This pattern can include letters, numbers, or enumerated values. For example, you could have revisions labeled 1st Rev, A, or 1.

To define a scheme for labeling revisions, you must build a revision sequence. This sequence specifies how objects should be labeled, the type of label to be used, and the number of revisions allowed. When you create a revision sequence, use the following syntax rules:

Rule	Example
Hyphens denote range.	A-Z signifies that all letters from A through Z inclusive are to be used.
Commas separate enumerated types.	Rev1,Rev2,Rev3,Rev4 is a sequence with four revision labels. Rev1 will be assigned before Rev2, which will be assigned before Rev3, and so on.
Square brackets are used for repeating alphabetic sequences.	[A-Z] signifies that the sequence will repeat after Z is reached. When it repeats, it returns to the front of the label list and doubles the labels so that the next sequence is AA, AB, AC, and so on.
Rounded brackets are used for repeating numeric sequences	(0-9) signifies a regular counting sequence. (When 9 is reached it will repeat and add a 1 before the symbol, etc.).
A trailing ellipsis (...) means a continuing sequence	A,B,C,... signifies the same thing as A-Z. 0,1,2,... signifies the same thing as (0-9)

These rules offer flexibility in defining the revision labeling sequence. Although you cannot have two repeating sequences in a single definition, you can include combinations of enumerated values and ranges within a repeating sequence. For example, the following revision sequence definition specifies that the first object should be labeled with a hyphen and the first revision should be labeled I, the second II, the third III, etc. After the fifth revision, all revisions will have numeric sequencing.

```
-,I,II,III,IV,V,(0-9)
```

The following policy definition uses an enumerated revision sequence:

```
Unrevised,1st Rev,2nd Rev,3rd Rev,4th Rev,5th Rev
```

If your location requires a numeric value, for example, for pre-released revisions, and then an alphanumeric scheme after that, the approach should be to change the policy at the point when the revision scheme should change. A separate policy is created and applied to a new revision, providing different states, signatures, etc. as well as a different revision scheme. For example, if a revision sequence is defined as:

```
0,1,2,...,-,[A,B,C,D,E,F,G,H,J,K,L,M,N,P,R,T,U,V,W,Y]
```

the automatic sequencing will never get beyond the number counting, so the entries after that are ignored. Two policies should be established for the object type with revision sequences defined as follows:

```
0,1,2,...
```

and

- , [A,B,C,D,E,F,G,H,J,K,L,M,N,P,R,T,U,V,W,Y]

If you want to exclude certain letters (such as I, O, Q, S, X, and Z in above), you must indicate only those you want to include as above. Use of a sequence such as [A-H,J-N] skips the letter I when automatically entering the revision during object creation, but does not prevent manually entering it during object creation or object modification.

If you enter blank spaces within the definition of a revision sequence, Live Collaboration uses the blank spaces literally. (In general, you should NOT use blank spaces within a revision sequence.) Consider the following examples:

Enter the revision sequence as:	Live Collaboration recognizes this as:
A, B, C	“A” “ B” “ C”
A,B,C,	“A” “B” “C”
1st Rev, 2nd Rev, 3rd Rev	“1st Rev” “ 2nd Rev” “ 3rd Rev”
1st Rev,2nd Rev,3rd Rev	“1st Rev” “2nd Rev” “3rd Rev”

Assigning a Store

The store identifies where files that are checked in under this policy will be stored by default. Stores are defined in MQL. If you intend to associate files with business objects governed by this policy, you must include the store in the policy definition. For information on changing the store for a policy, see *Changing the store for a policy*.

When using a 3DEXPERIENCE product to check in a file, the person or company default store is used regardless of the store set by the policy.

For example, assume you have a policy for proposing and presenting drawings for review. These drawings may be of various types and formats. However, all the drawing files can be contained in one file store called Drawings. This file store identifies where the drawing files will be stored. Consult your System Administrator for your options. When a file is checked into a business object governed by this policy, that file will be stored according to the definition of the Drawings file store.

To assign a store to the policy

- Type the store name in the **Store** text box.

  Or
- Click the **Store** ellipsis button and select a store from the Store Chooser that appears.

MQL users and programmers can override the store specified in the policy, by including the store in the checkin command.

Defining the Hidden Option

You can mark the new policy as “hidden” so that it does not appear in the Policy chooser. You may want to use the hidden option if, for example, an object is under development or if it is intended only for your personal use. Users who are aware of the hidden policy’s existence can enter its name manually where appropriate. Hidden objects are also accessible through MQL.

Selecting Object Types to be Governed

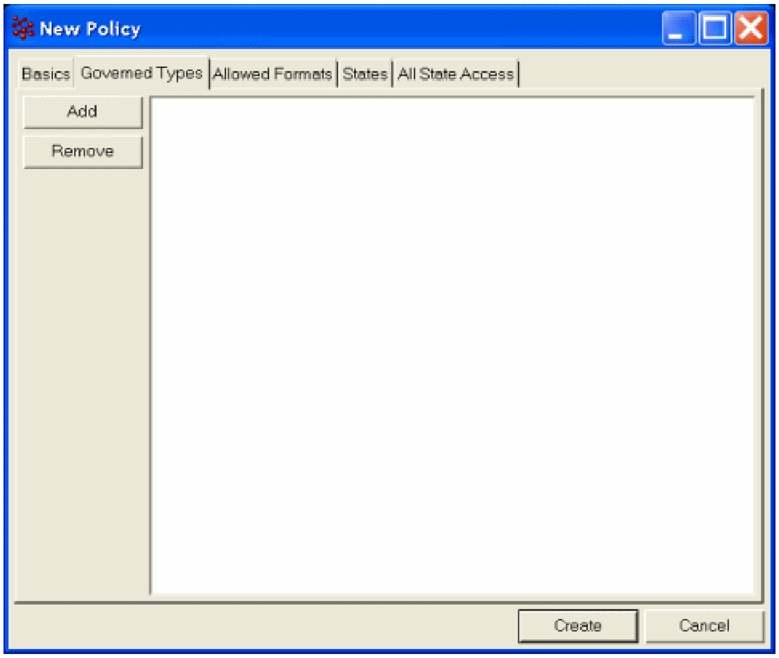
Just as a policy may govern many different object types, each object type may have many different policies that can govern it. The difference, though, lies in that an object can only have one policy associated at any time.

For example, assume you have an object type named “Drawing.” This type may be governed by two policies named “Engineering Drawing Process” and “Documentation Drawing Process.” When an object of type “Drawing” is created, you must indicate the policy that will govern the object instance. A drawing meant for documentation will have a different review and lifecycle from a drawing that is a component to an engineering assembly. By associating either the “Engineering Drawing Process” or the “Documentation Drawing Process” policy with the created object, you are controlling the types of files that can be checked in, who will use the object, and when it will be used.

Policies can also govern the child types of any type added to its definition. For more information on inherited types, see *Assigning a Parent Type* in Chapter 4.

To specify the governed types

1. Click the **Governed Types** tab in the New Policy dialog box. The following dialog box opens:



2. Click **Add**.

   The Type Chooser opens.

3. Select one or more types. Use **Ctrl+Click** to select multiple types.

4. Click **OK**.

The selected types are listed in the Governed Types dialog box.

To remove a governed type from the policy definition

1. In the New Policy dialog box, click the **Governed Types** tab.
2. Select the type you want to remove.
3. Click **Remove**.

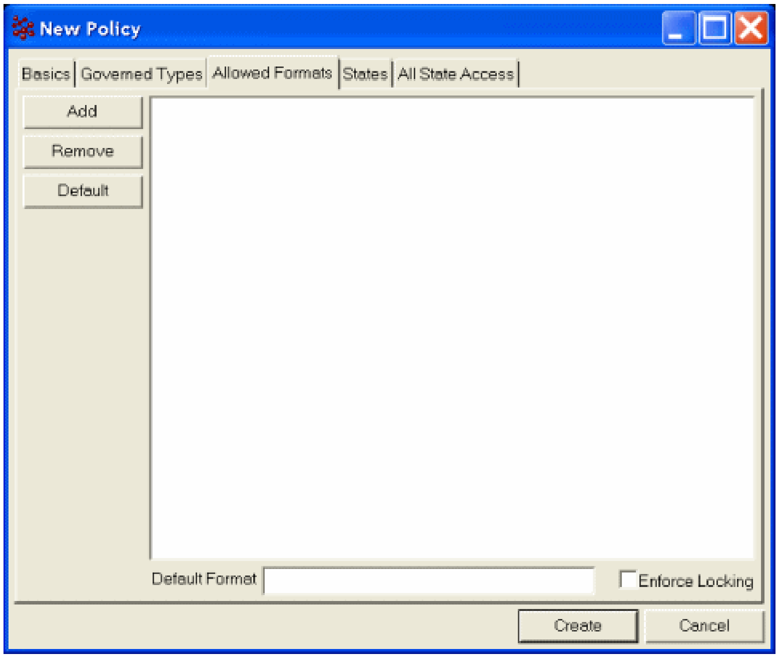
Selecting a Format

Depending on the policy and the object being created, only certain files formats are appropriate. Controlling the formats permitted allows you to restrict the types of files that a user can associate with a business object.

For example, you could have a policy that governs expense reports. This policy would need formats for processing files that contain financial and other expense report information. In this case, you might specify that two file formats are allowed: Lotus 1-2-3 and Excel. As described in *Overview* in Chapter 7, formats specify the programs required to view, edit, and print files.

To specify formats

1. Click the **Allowed Formats** tab in the New Policy dialog box.



2. Click **Add**.

   The Format Chooser opens.

3. Select one or more formats. Use Ctrl-click to select multiple formats.
4. Click **OK**.

   The selected formats are listed in the **Allowed Formats** dialog.

To remove a format from the policy definition

1. In the New Policy dialog box, click the **Allowed Formats** tab.

2. Select the format you want to remove.

3. Click **Remove**.

Defining a Default Format

The default format is used when a user selects the View, Edit or Print options for an object. When you select formats, as described above, the first format that you select is recognized as the default. The format name is displayed in the **Default Format** text box.

To select a different default format

1. In the New Policy dialog box, click the **Allowed Formats** tab.

2. Select a format.

3. Click **Default**.

The format name is displayed in the **Default Format** text box. If an object does not have any files checked into its default format, execution of a View, Edit, or Print command will check its other formats and open files found there.

Enforcing Locking

Check **Enforce Locking** to prevent one user from overwriting changes to a file made by another user. When an object is governed by a policy that has enforce locking turned on, the only time a user can check in files *that replace existing files* is when:

- the object is locked

  and

- the user performing the checkin is the locker

To ensure that the person who locked the object is the person who checked out the file, enforce locking disables the manual lock function (choosing Lock from the Files menu). The only way to lock an object that is governed by a policy that enforces locking is by checking Lock Object when checking out the file.

When checking in files *that do not replace existing files* (for example, if you check files into a format that contains no files or you append files), as long as the object is unlocked, you can check in new files.

Enforce locking ensures that when a user checks out a file and locks the object, signifying that the user intends to edit the file, no other user can check in a file and overwrite the files the original user is working on. When the original user checks the file back in, the user should unlock the object.

When an object is locked, no files can be checked into the object until the lock is released, even if the file does not replace the checked-out file that initiated the lock. This means that attempts to open for editing, as well as checkin, will fail. Files can be checked out of a locked object and also opened for viewing.

Be aware that the manual unlock command (selecting Unlock from the Files menu) is available for users who have unlock access, but users should avoid using the command for objects that have enforce locking. For example, suppose Janet checks out a file and locks the object with the intention of editing the file and checking it back in. Steve, who has unlock access, decides he needs to check in an additional file for the object so he unlocks the object manually. When Janet attempts to check in her edited file, replacing the original with her updated file, the system won’t allow her to because the object isn’t locked. In order to check in the file, Janet has several options:

- she can check in the edited file in such a way that it won’t replace existing files; for example, change the name of the edited file and append it or delete the original file and check in the edited file
- she can check out the original file again and lock the object, taking care not to replace the edited file on her hard drive with the older file she is checking out, and then check in the edited file

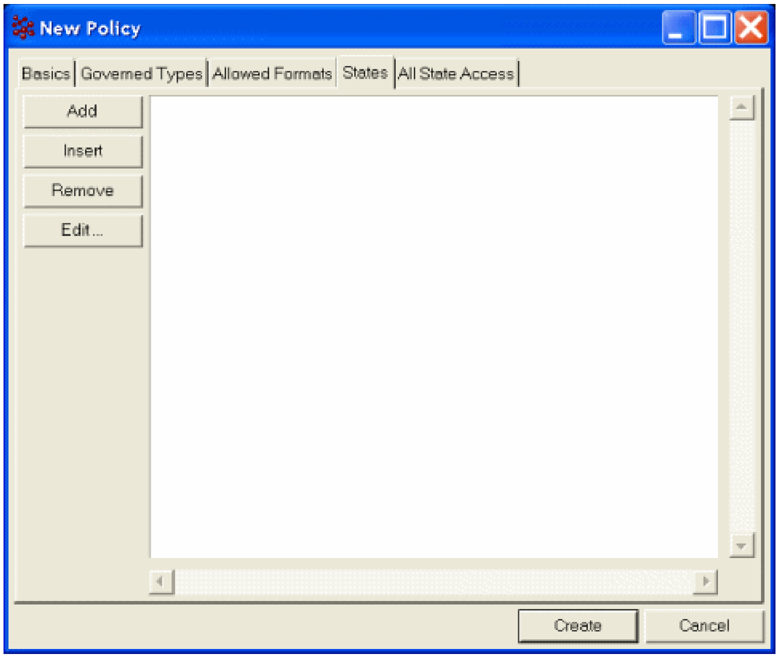
For more information on unlock access, see Live Collaboration - Administration>People and Organizations>Controlling Access>User Access>Accesses>Unlock Access.

A user would end up in a situation similar to the one described above if the user forgets to lock the object when checking out a file for editing. When the user attempts to check in the edited file, the system won’t allow the checkin because the object is unlocked (or possibly locked by another user who checked out the file after the first user).

Working with States

The **States** tab of the New Policy dialog box enables you to add states to the lifecycle defined in a policy. You also can select a state for editing or removal.

To work with states, click the **States** tab in the New Policy dialog box.



Adding or Inserting a State

You can add a state at the end of the lifecycle list or insert one at a specific point in the lifecycle.

To add a state at the end of the lifecycle

1. Click **Add**.

   An "Untitled" state is displayed, indicating that the new state is undefined.

2. Select the new state.
3. Click **Edit**.

   The Edit State dialog box opens.

To insert a state at a specific point in the lifecycle

1. From the **States** tab in the New Policy dialog box, select an existing state.

   The state that you will insert will appear *before* this selected state.

2. Click **Insert**.

   An "Untitled" state is displayed, indicating that the new state is undefined.

3. Select the new state.
4. Click **Edit**.

The Edit State dialog box opens, showing the Basics tab.

This dialog box enables you to define information such as who can access a business object, what type of access a user can have, whether or not new versions or revisions are allowed, whether the object can be automatically promoted, and the notifications that will be made when the object reaches this state.

5. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

You can specify the name of the state you are creating. All states must have a unique state name within this policy. You should assign a name that has meaning to both you and the user. The state name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

For example, assume you have a process for performing and evaluating lab tests. The first state might involve receiving the initial test request, gathering information on the item or person to be tested, and getting approval for the test. This state could be called “Initial Test Processing” or “Test Request.”

Once the testing is approved, the test object might enter a second state where the test is actually performed. This state could be called the “Testing,” “Actual Test Processing,” or “Lab Work” state.

After the test is complete, the object might then be available for evaluation and review. This final state could be called the “Test Results,” “Test Evaluation,” or “Test Review” state.

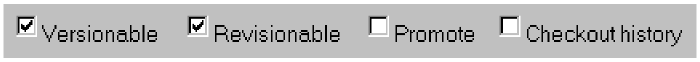
In each example, the names provide some indication of what is happening to the test object in each state.

Assigning an Icon

You can assign a special icon to the new state. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described on *Defining an Icon* in Chapter 1.

Setting the Versionable and Revisionable Buttons

Check the **Versionable** option on the Edit State dialog box to allow a file to be checked into the object when it is in this state. If **Versionable** is not set, check-in is not allowed.

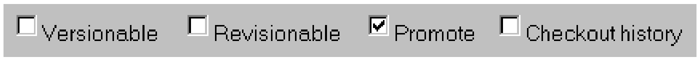


☑ Versionable ☑ Revisionable □ Promote □ Checkout history

Check the **Revisionable** option on the Edit State dialog box to specify whether or not revisions of the object can be created while the object is in this state. If **Revisionable** is not checked, no new revisions are allowed in this state. For example, assume you have a state named “Tax Return Completed.” In this state, you do not want anyone to revise the objects containing the finished tax returns. In this case, you would not check **Revisionable**.

Selecting Automatic Promotion

Check the **Promote** option on the Edit State dialog box to allow the object to be promoted to the next state automatically when a signature is modified and all requirements are met.



☐ Versionable ☐ Revisionable ☑ Promote ☐ Checkout history

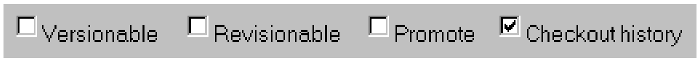
With this box checked, when a signature is approved or ignored, Live Collaboration will try to promote the business object. If all signature and check requirements are satisfied, Live Collaboration will promote the business object automatically. If there are no signatures or check requirements on a state, this setting has no meaning.

If there are no requirements on a State, the promote subclause has no meaning.

Selecting Checkout History

Check the **Checkout History** option on the Edit State dialog box to allow the event to be recorded when a file is checked out. The generation of history information on the checkout event is optional. The need to disable checkout history stems from the implementation of distributed databases and the advanced search partial index process. Creating history records requires that a distributed transaction be run across multiple servers. If any server is unavailable, the transaction will fail. This means that all servers must be available in order to checkout/view files. If checkout history is disabled, only the local server needs to be accessible in order for the transaction to run to completion.

During a partial index process, business objects are indexed according to their modification date. If checkout history enabled, the history record and the modification date of the business object is updated whenever a file is checked out. Since the modification date is changed, during a partial index process, the business object and its associated files will be indexed again even if the business object has not changed except for the file check out. Disabling checkout may beneficial in preventing a lot of unnecessary indexing, especially if large files are often checked out.

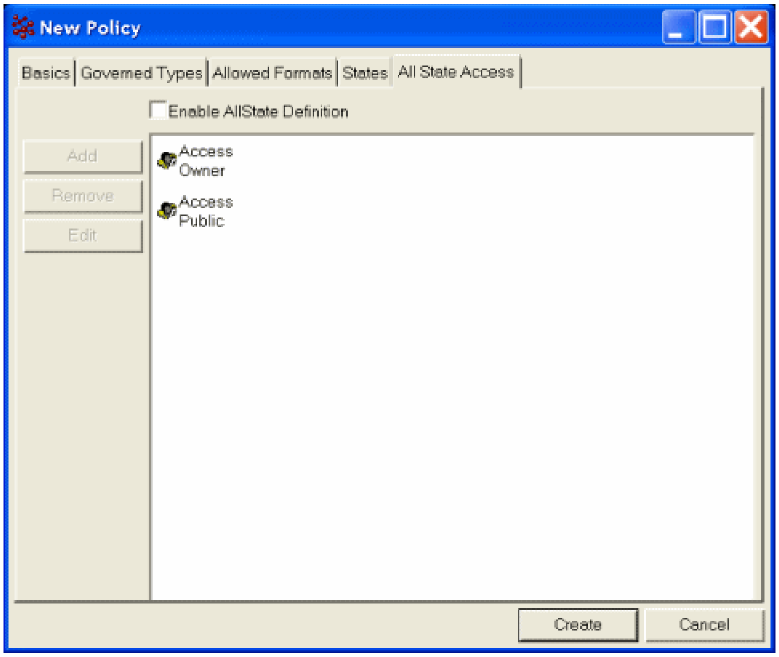


☐ Versionable ☐ Revisionable ☐ Promote ☑ Checkout history

Assigning All-State Access

The **All State Access** tab of the New Policy dialog box enables you to define an all state access rule that applies to every state in a policy. These rules grant access in addition to any access rules defined for a specific state.

IMPORTANT: When adding or modifying accesses, use MQL. There are settings available in MQL that are not available in Business Modeler, such as those used to configure modern access security used by apps.



□ Enable AllState Definition

To assign all state access for a policy

To define access across all states in a new policy, check **Enable AllState Definition** and assign access as described in *Assigning Access*. Rather than specifing who will have access to a business object in a particular state, the access you give will be applied to all states in that policy.

You can also modify an existing policy to add or remove all state access by checking or un-checking **Enable AllState Definition**.

Assigning Access

The **Access** tab on the Edit State dialog box enables you to specify who will have access to a business object in this state and how much access will be allowed.

As stated in the overview of this chapter, there are three categories of access: **Public**, **Owner**, and **User**. Public refers to everyone in the database, owner refers to the person who creates the business object or to whom the object was routed or reassigned, and user can be defined as any person, group, role, or association. In the **Access** area, you edit an access item to define the access.

IMPORTANT: When adding or modifying accesses, use MQL. There are settings available in MQL that are not available in Business Modeler, such as those used to configure modern access security used by apps.

Using a Filter Expression

User access lists defined on a policy state can accept a filter expression in order to grant or deny access to a specific user. If the filter expression evaluates to “true,” the specified access will be granted; otherwise the access is denied. This provides an additional level of access granularity, which increases security and reduces overall management problems by reducing the number of policies or rules required.

Expression access filters can be any expression that is valid in the Live Collaboration system. Expressions are supported in various modules of the Live Collaboration system, for example, query where clauses, expand, filters defined on workspace objects such as cues, tips and filters, etc.

Since you must have at least read access to the business object in order to evaluate the filter, normal access checks must be disabled while an access filter is being evaluated.

Policies and rules should use organization, project, maturity, and reserve logic instead of access filters whenever possible. These security checks ensure the completeness of search results. You can also use the Configure My ENOVIA app for defining P&O.

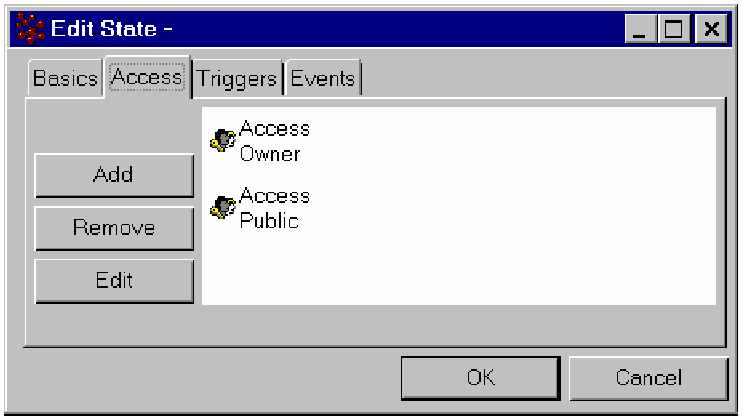
Attribute values that come from interfaces cannot be used in a create access rule, because the interfaces are not applied until AFTER the create access checks.

To assign access for a policy state

1. Click the **Access** tab in the Edit State dialog box.

   The Owner and Public access categories are already listed. A state always has one Public access and one Owner access item. However, User access items can be quite numerous. For example, you might want to specify different levels of access for different user categories—persons, groups, roles, and associations.

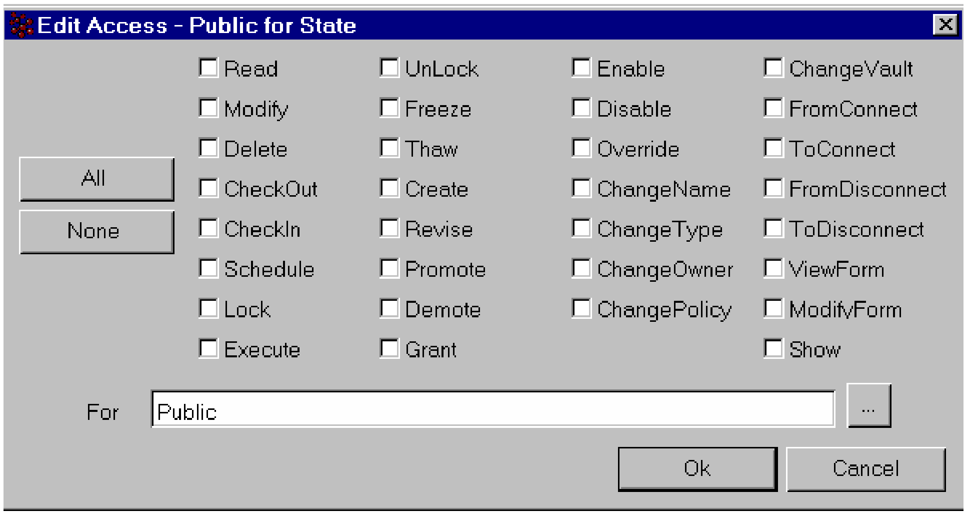
   For descriptions of all of the accesses, see the *Administration Guide : Accesses* in the online documentation.



2. Define the accesses for the public.

   a) Select the **Public Access** item and click **Edit**.

   The Edit Access dialog box opens.



For Public

b) Check the accesses you want all users in the database to have for the current state. The public should typically have few accesses, such as read and checkout.

c) Click **OK**.

3. Define the accesses for the owner.

a) Select the **Owner Access** item and click **Edit**.

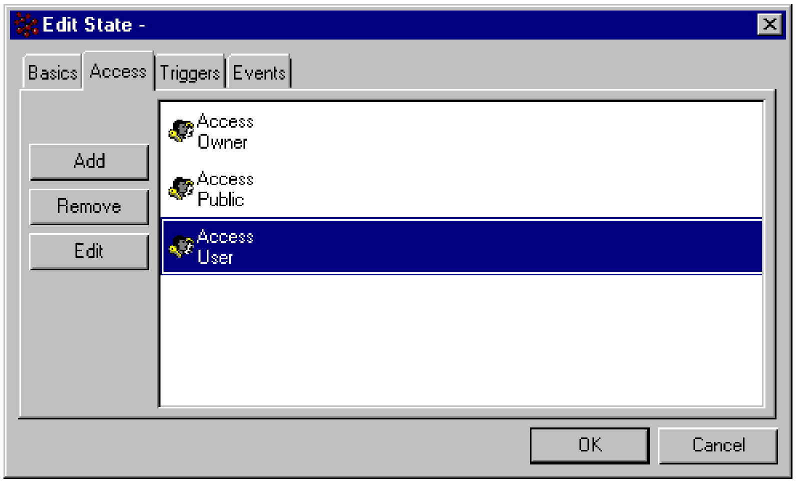
b) Check the accesses you want the object owner to have for the current state. Owners typically have full access.

c) Click **OK**.

4. Define user accesses.

a) On the Access tab in the Edit State dialog, click **Add**.

A User item is added.



Access Owner

Access Public

Access User

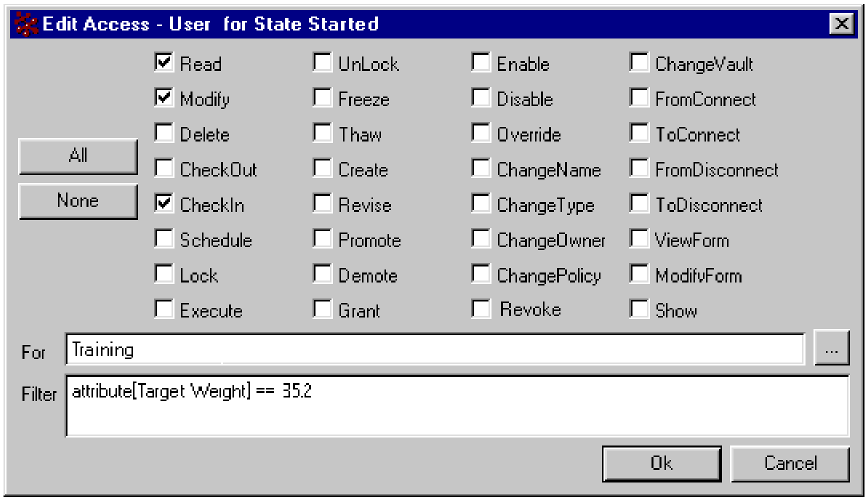
b) Select the **User Access** item and click **Edit**.

c) Click the ellipsis button to the right of the For text box.

d) In the User Chooser, select the person, group, role, or association for which you want to assign access.

e) Check the accesses you want to assign to the user.

f) Optionally, add a filter in the **Filter** text box. (See *Using a Filter Expression* for details.) When the expression evaluates to “true,” the specified access is granted. The following is a example of how the user access could be set up in a state:



☑ Read	☐ UnLock	☐ Enable	☐ ChangeVault
☑ Modify	☐ Freeze	☐ Disable	☐ FromConnect
☐ Delete	☐ Thaw	☐ Override	☐ ToConnect
☐ CheckOut	☐ Create	☐ ChangeName	☐ FromDisconnect
☑ CheckIn	☐ Revise	☐ ChangeType	☐ ToDisconnect
☐ Schedule	☐ Promote	☐ ChangeOwner	☐ ViewForm
☐ Lock	☐ Demote	☐ ChangePolicy	☐ ModifyForm
☐ Execute	☐ Grant	☐ Revoke	☐ Show

For: Training

Filter: attribute[Target Weight] == 35.2

Here a filter Expression is being applied to the User Access called “Training” in the state “Started.”

Without the expression access filter, the selected access (read, modify and checkin—see the screen shot above) would have been allowed:

- if context user belongs to the group “Training” and
- business object being accessed is in the state “Started”

With the expression access filter, the selected access (read, modify, checkin) would be allowed only:

- if context user belongs to the group “Training” and
- the business object being accessed is in the state “Started” and
- if filter expression (attribute[Target Weight] == 35.2) is evaluated as true for the business object being accessed.

Note that in this situation “Target Weight” is an attribute of the business object being accessed. If the “Target Weight” attribute doesn’t exist for the business object being accessed, the read, modify and checkin access would not be allowed for the context user via this user access.

g) Click **OK**.

To remove a user access item

1. Select the User item.
2. Click **Remove**.

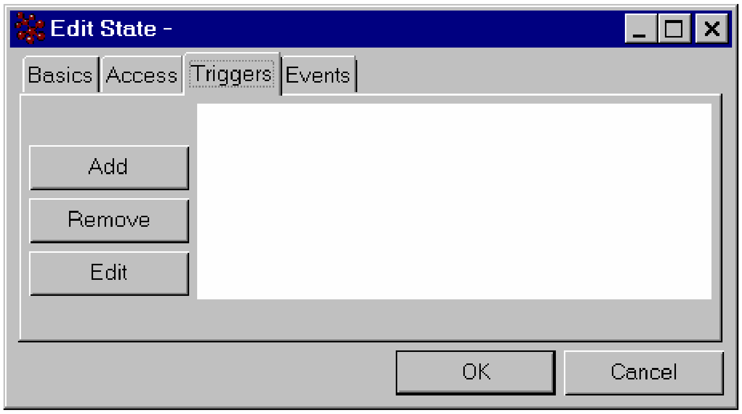
Note that you cannot add or remove the Owner Access or Public Access items.

Assigning Event Triggers

Event Triggers provide a way to customize Live Collaboration behavior through Program Objects. Triggers can contain up to three Programs, (a check, an override, and an action program) which can all work together, or each work alone. Many business object events are supported for lifecycle triggers. For more information, see the *Configuration Guide : About Triggers*.

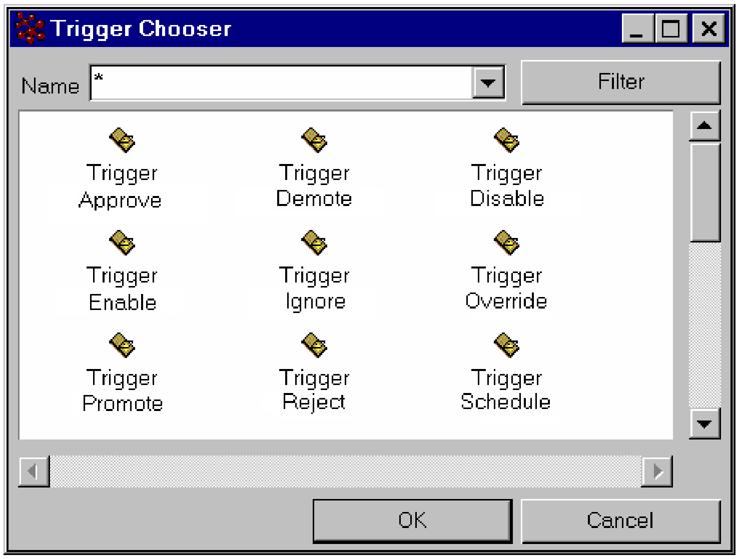
To assign event triggers

1. Click the **Triggers** tab in the Edit State dialog box. The following dialog opens:



2. Click **Add**.

   The Trigger Chooser opens.



   The Trigger Chooser contains Trigger Objects for each of the legal events available for States.

3. Select one or more triggers and click **OK**. Use Ctrl-click to select multiple triggers.

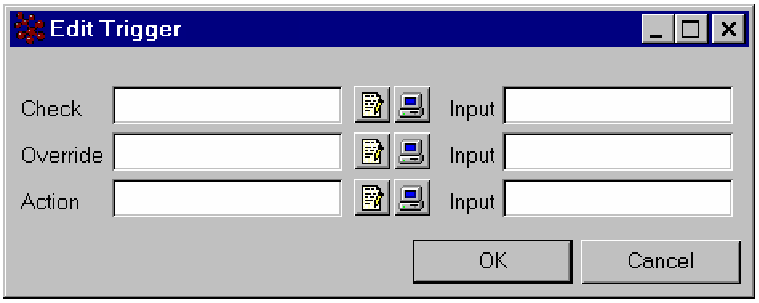
   The selected triggers are listed in the **Triggers** dialog.

After assigning a trigger to the state, you must assign the appropriate programs to each of the trigger types.

To assign programs to a trigger

1. Double-click the trigger icon.

   The Edit Trigger dialog opens.



Each supported data event has three types of triggers:

Checka trigger that executes before the event occurs.

Overridea trigger that can replace the event transaction.

Actiona trigger that executes after the event occurs.

For the selected event, you can specify programs for none, any, or all of these trigger types.

2. Type the name of the program to associate with the trigger in the text box corresponding to the type of trigger.

   Or

   Click 🖥 to display the Program Chooser dialog box and select the program that should execute when the event occurs.

   To edit the program named in the text box, click 📝 to display the Edit Program dialog box.

3. In the **Input** field to the right of the **Check**, **Override**, or **Action** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

   Environment variable “0” always holds the program name and is set automatically by the system.

   Arguments from the **Input** field are set in environment variables “1”, “2”, . . . etc.

4. Repeat Steps 2 and 3 for each trigger type.

5. When all necessary trigger programs have been assigned, click **OK** to create the event trigger.

To remove an assigned trigger

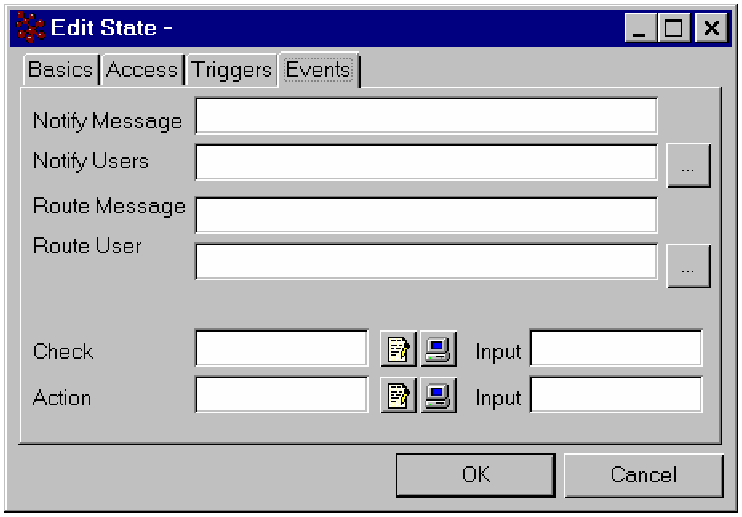
1. In the Edit State dialog box, click the **Triggers** tab.
2. Select the trigger you want to remove.
3. Click **Remove**.

Defining Events

Events include Notify and Route messages, also Check and Action Programs.

To define events

1. Click the **Events** tab in the Edit State dialog box.



2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Notification Message

You can specify a message to be sent to selected users once the business object has entered the state you are defining. This message can notify users that an object is now ready for some particular action. The notification message is limited to 255 characters.

Depending on the settings in each person definition, each recipient will receive the message in IconMail, email, or both.

To define a notification message

1. Type the message in the **Notify Message** text box.
2. If you know the exact names of the users to whom the message should be sent, type them in the **Notify Users** text box.

   Or

   Click the **Notify Users** ellipsis button and select the names of the persons, groups, and roles who should be notified with the message from the User Chooser that appears.

For example, assume you have a user manual that is being written. In its beginning state, only the author and the author’s manager might access the document. However, once the manual is ready for review, it would most likely be promoted to a state where it is available to other users for comments. When this occurs, the users must be notified that the manual is available and that their review comments are required. You might send a message such as:

```
The User’s Guide is now ready for review. Please have your
review comments completed in two weeks.
```

Defining a Route Message

You can automatically reassign ownership of the object when it reaches the state that you are defining. At the same time, you can send a message to the users who will receive ownership. The route message is limited to 255 characters.

To define a route message

1. Type the message in the **Route Message** text box.
2. If you know the exact name of the user to whom the message should be sent, type it in the **Route User** text box.

   Or

   Click the **Route Users** ellipsis button and select the name of the person, group, or role who should receive ownership of the object and be notified with the message from the User Chooser that appears.

For example, assume you have a manual that was just completed by a writer. That manual is promoted into a state called “Formatting and Editing” in which an editor takes complete charge of the manual. That editor prepares the document for review and oversees any changes required to prepare the manual for publication. Since the writer is no longer involved, you may want to assign the manual’s ownership to the editor. While the editor might be among the users notified that the manual is finished, the editor should receive special notification that the manual now “belongs” to him/her. After the manual is published, you might again change the ownership to that of the company librarian. When changes in ownership occur, the new owner should be notified.

Associating a Check Procedure With Promotion of a State

The **Check** text box on the Edit State dialog box enables you to define a verification procedure to be associated with the promotion of an object to the next state.

The procedure is specified as a program which is executed when a person tries to promote the object. When executed, the procedure returns a true or false value. If the value is true, the object can be promoted. If the value is false, the promotion is denied.

Policies can employ Checks as defined before Event Triggers were available. It is advisable, however, that new Checks are attached as Triggers, since greater functionality is available there. For more information, see the *Configuration Guide* : *About Triggers*.

To associate a check procedure with promotion

1. If you know the exact name of the program to be executed, type it in the **Check** text box.

   Or

   Click 🖥 to display the Program Chooser and select the program that should execute when the event occurs.

   To edit the program named in the text box, click 📝 to display the Edit Program dialog box.
2. In the **Input** field to the right of the **Check** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

   Environment variable “0” always holds the program name and is set automatically by the system.

   Arguments from the **Input** field are set in environment variables “1”, “2”, . . . etc.

Refer to *Overview of Programs* in Chapter 11, for information on creating the Program Object.

Associating an Action With Arrival in a State

The **Action** text box on the Edit State dialog box enables you to automate an activity upon arrival in a state. Once an object is promoted to this state, Live Collaboration executes the program specified. This entry is useful, for example, to execute procedures that might notify non-Live Collaboration users or place orders for equipment or services.

Policies may employ Actions as defined before Event Triggers were available. It is advisable, however, that new Actions are attached as Triggers, since greater functionality is available there. For more information, see the *Configuration Guide : About Triggers*.

To associate an action with arrival in a state

1. If you know the exact name of the program to be executed, type it in the **Action** text box.

   Or

   Click 🖥 to display the Program Chooser and select the program to be executed.

   To edit the program named in the text box, click 📝 to display the Edit Program dialog box.

2. In the **Input** field to the right of the **Action** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

   Environment variable “0” always holds the program name and is set automatically by the system.

   Arguments from the **Input** field are set in environment variables “1”, “2”,. . . etc.

Refer to *Overview of Programs* in Chapter 11, for information on creating the Program Object.

Creating the State Definitions

After you have entered all appropriate information on the Edit State dialog box, click **OK** to create the definitions. You are returned to the New Policy dialog box.

Removing a State Definition

To remove a state definition

1. In the New Policy dialog box, click the **States** tab.
2. Select the state you want to remove.
3. Click **Remove**.

Removing a State From a Policy That is in Use

Removing a state from a policy that is governing objects is not recommended. An alternate approach is to clone the policy and then remove the state from the clone. There is the notion that “from this point on” the policy will control these types of objects. New objects should use the new policy and older objects can change to the new policy, if desired.

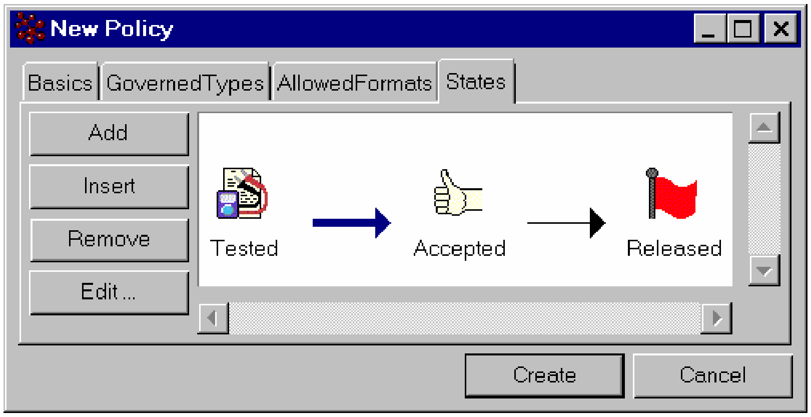
When a state is removed from a policy, all signatures to and from the state are removed. If the policy is in use, all signature approvals, comments, etc. are deleted.

Assigning Signature Requirements

A signature specifies who can control the promotion or rejection of a business object. When an object is promoted, it moves to the next defined state and is subject to the accesses assigned for that new state. When an object is rejected, it remains in the current state until it meets the criteria for promotion.

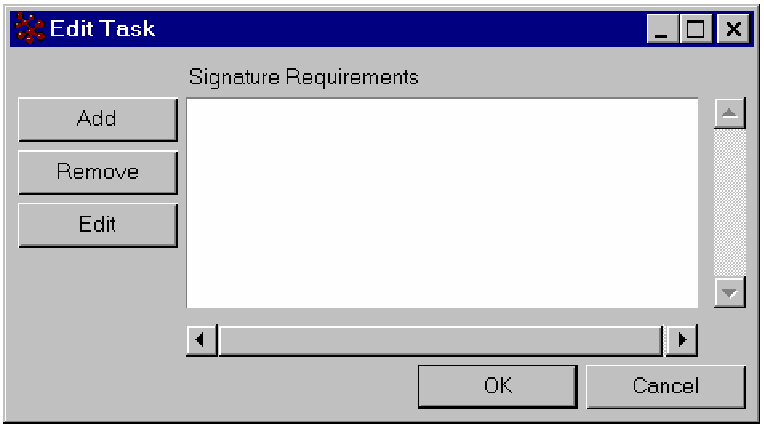
To assign signature requirements

1. Select a transition arrow between states on the Policy dialog box.
   For example:

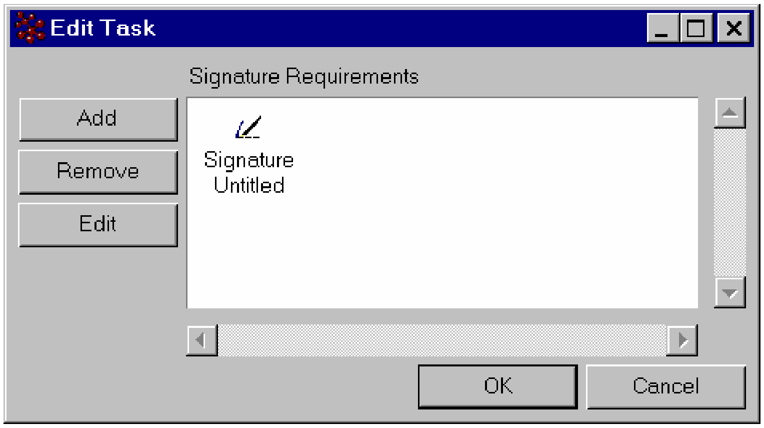


2. Click **Edit**.

   The Edit Task dialog box opens.



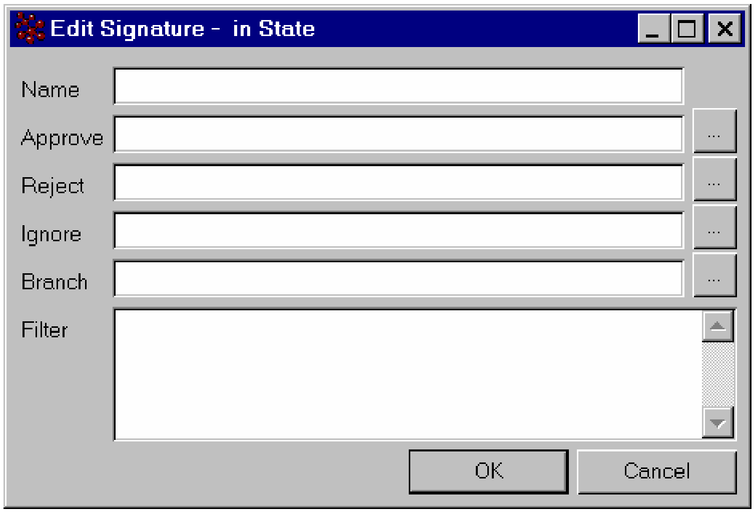
3. Click **Add**. An “Untitled” signature is displayed. This new signature is undefined.



4. Select the new signature.

5. Click **Edit**.

   The Edit Signature dialog box opens.



6. Use this dialog box to define the purpose of the signature, optional branching, and who will have control over the promotion of the object from the state you are defining. These are described in the following sections.

Defining the Name

All signatures in a policy must have a unique signature name assigned. You should assign a name that has meaning to both you and the user. The signature name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

It is best to use a name that identifies what the signature represents. For example, the signature might represent a group which is required to sign off (such as Facilities Planning) or a role which needs to approve (such as Safety Engineer) or a specific person (such as Mary Smith).

Defining Who Can Approve, Reject, and Ignore

The next three areas of the Edit Signature dialog box specify who has access to approve, reject, or ignore a signature.

- The **Approve** text box specifies who can sign for an approval.
- The **Reject** text box specifies who can sign for a rejection.
- The **Ignore** text box specifies who can satisfy the signature requirement without approving or rejecting. When an Ignore signature is specified, the named user can sign in the place of others. For example, this signature is used when you want to offer a more senior manager the ability to sign for a lower-level manager, or when one or more of the signatures may not be required in all circumstances.

Note the difference between Override Access and Ignore Signature Access (and refer also to the section *Assigning Access*):

- Override Access is allowed or not allowed on the state. The signature and check requirements are not necessarily satisfied, but the object can be promoted anyway.
- Ignore Signature Access assumes that a signature is required for object promotion. It allows another user to sign in place of a user, thereby satisfying the signature requirement.

For example, assume you have a state named “Planned.” In this state, objects containing proposed projects are created and assigned. While the object is in this state, you might require two signatures in order to promote it to the next state.

The first signature could be named Contracted to indicate when a contract to complete the job has been signed by an outsource company. The second signature could be named Assigned to indicate when an internal owner of the project has been assigned. In this case, either one signature or the other would be appropriate: the project would be completed by either an internal employee or an external source.

The Assigned signature could allow the manager of the project approve, reject, and ignore privileges. The manager would approve once s/he assigned it to an employee. When defining the signature Contracted, you could include in the Approve, Reject, and Ignore text boxes the group or role responsible for negotiating and signing contracts. (Assigning a role or a group rather than a specific user allows the policy to be more generic for use in more circumstances.) In the Ignore text box, you could also include the role defined in the Assigned signature. This role could then “ignore” the Contracted signature, fulfilling the promotion requirements. The object could then be promoted to the next state without involving another user.

State Branching

A *branch* defines what the next state will be after a signature is applied. For each state, it is possible to have more than one branch. Which branch is taken depends on which signature is satisfied.

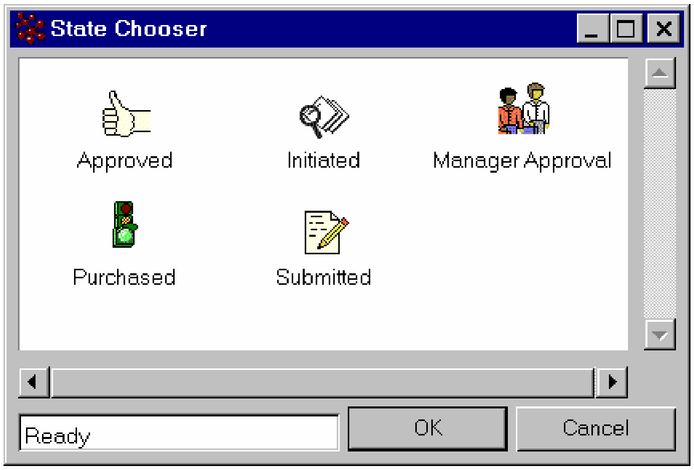
In the **Branch** text box, you provide a reference to another state. If none is specified, the default is the ‘next’ state as determined by state insertion order. By specifying a branch, you can decide which signatures are required to transition to a given state during a promote operation. When promotion is initiated, the system will choose the state for which all signatures are satisfied. If more than one branch is enabled, an error is generated.

To enable state branching

1. In the **Branch** text box, type the name of the state where you want to branch.

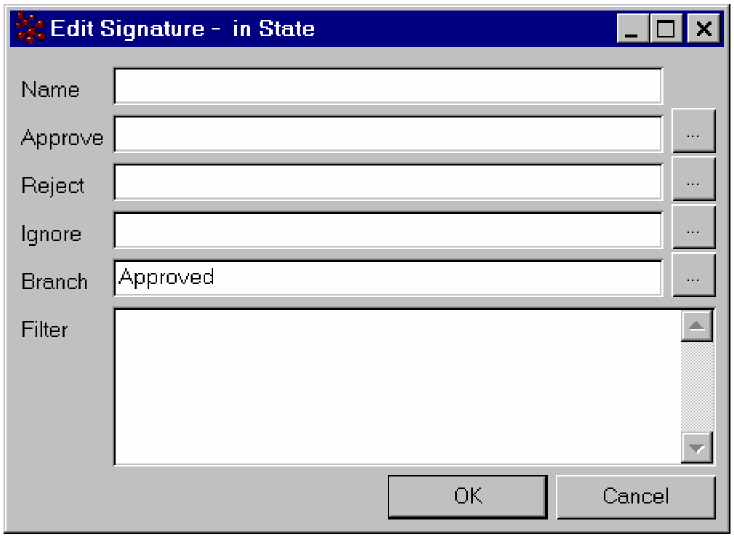
Or

Click on the ellipsis button next to the **Branch** field. The State Chooser opens, showing all states in the policy:



2. Select the state where you want to branch and click **OK**.

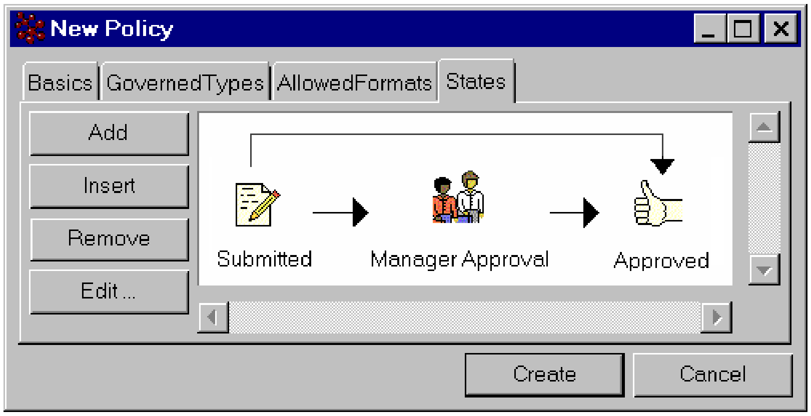
The display returns to the Edit Signature dialog box, showing the name of the selected state in the **Branch** text box:



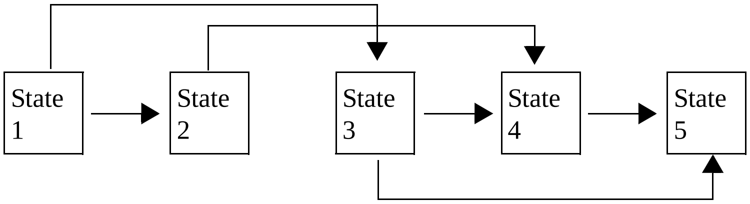
Branch: Approved

3. Click **OK**.

After the branch is set, the states area of the Edit dialog box graphically displays the branch with an arrow bypassing the skipped state(s).



States can have multiple branches, each enabled by a separate signature. Whichever signature is signed indicates which branch is taken. In this case, the States area in the Edit Signature dialog box would display all possible branches. For example:



Filters

The **Filter** is an expression that evaluates to either true or false. If a signature requirement *filter* evaluates to true, then the signature requirement is fulfilled. This is useful for adding required signatures that are conditional, dependent on some characteristic of a business object. The default rule is:

```
    current.signature[NAME].satisfied
```

which is a select field that means the signature has been approved, ignored, or overridden. When you specify a filter, this default rule is replaced.

Approval can become dependent on any selectable field of the business object. This includes attributes as well as states and other signatures. The real power of filters comes from the use of combinatorial logic using and’s and or’s between state information and business object information.

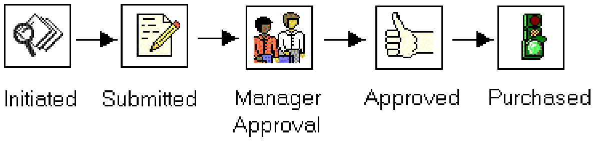
To specify a filter

1. In the **Filter** area, type the select expression which will cause the signature to evaluate to true.
2. Click **OK**.

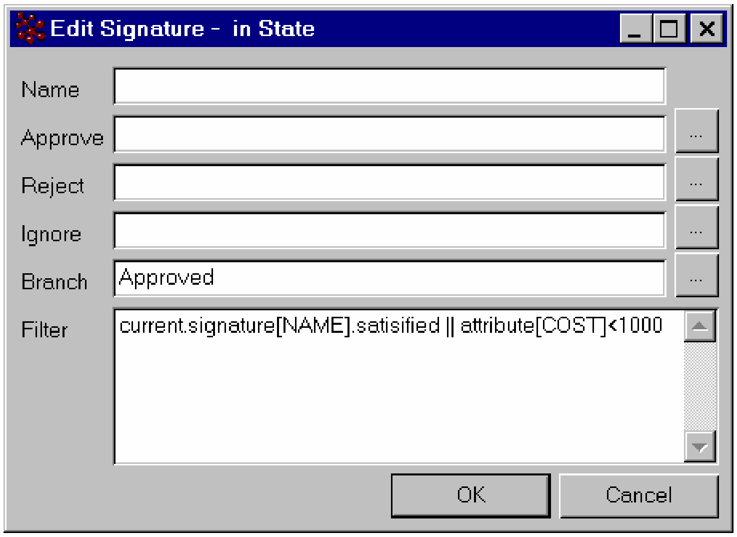
For help formatting the expressions that you can enter into th eFilter area, see the *Configuration Guide : Appendix: Selectables.*

To combine branches and filters

You can use a filter to define when a branch will take place. For example, a Purchase Requisition policy has five states:



The policy may require an additional signature if the Cost attribute of the object is over $1000. For purchase requisitions of $1000 or less, the Manager Approval state can be skipped, branching directly to the Approved state



Name

Approve

Reject

Ignore

Branch: Approved

Filter: current.signature[NAME].satisified || attribute[COST]<1000

The **Branch** text box would contain the state Approved and the **Filter** text box would contain the following expression:

```
current.signature[NAME].satisified || attribute[COST]<1000
```

which means that the signature is true when it is ‘satisfied’ (approved, ignored, or overridden) OR when the attribute value is less than 1000.

Removing a Signature Definition

To remove a signature, select the signature item in the **Signature Requirements** area then click **Remove**.

Completing a Signature Definition

Once you have completed all signature definitions, click **OK** on the Edit Task dialog box.

Completing a Policy Definition

After you have entered all of the appropriate information on the New Policy dialog box, click **Create** to create the policy.

Modifying or Deleting a Policy Definition

Modifying a Policy Definition

After you establish a policy definition, you can add or remove defining values. Refer to the section *Modifying an Administrative Object* in Chapter 1.

Changing policy states

If a state is added from the Edit Policy dialog box (rather than the New Policy dialog box), all object instances that are governed by this policy will be affected. (Refer to *Adding or Inserting a State*.) If the object is in a state that precedes the new state, a state is added, as desired, in the object's lifecycle. However, if the object's current state is beyond where the new state is added, the object will never be in that state except through demotion. In some cases, this is not a concern; but, states should be added to existing policies with care.

Removing a state from a policy that is governing objects is not recommended. An alternate approach is to clone the policy and then remove the state from the clone. There is the notion that "from this point on" the new policy will control these types of objects. New objects should use the new policy and older objects can change to the new policy, if desired.

Changing the store for a policy

Keep in mind that if you change the store for a policy, files that are already checked into objects governed by the policy will still reside in the old store. If these objects are revised or cloned, the new revision/clone inherits the original file and its storage location and thus the clone or revision will be placed in the old storage location. Any new files that are checked in will be placed in the new store.

However, the old store will still be used in the case where another object contains a reference to files in the object that now uses a different store. When the time comes for the reference to become an actual file (as when the file list changes between the 2 objects) the file copy is made in the same store the original file is located in.

Deleting a Policy

If a policy is no longer required, you can delete it. Refer to the section *Deleting an Administrative Object* in Chapter 1. However, Live Collaboration will not allow you to delete a policy if any object is governed by it. To delete a policy used by any object, you must reassign the object to another policy or delete the object before deleting the policy.

10

Working With Business Wizards

Introduction

The Business wizards feature provides you with the ability to create a user interface to simplify repetitive tasks performed by users.

A *wizard* is a program which asks the user questions and executes a task based on the information received. It consists of one or more dialog boxes, called *frames*, which are presented sequentially to the user. Generally, each frame provides some explanation or asks a question, then requires that the user either type a response or make a choice from a list. When all information has been collected, the wizard program performs an action based on the information.

The Business wizards feature allows you to create a wizard, choosing the number of frames and defining the contents of each frame to customize the wizard for a specific purpose relating to your database.

Suppose, for example, that a number of users are required to check files into the database on a weekly basis. A wizard could ensure that these files are named according to a standard naming convention, prevent the accidental replacement of existing files, and track which users have submitted a file, even sending IconMail or email to the manager who needs the files to advise that the they have been checked in.

Overview of Business Wizards

A wizard is a type of Program object. This means that a wizard can be launched most of the places that a Program can be launched, for example, stand-alone, or as a method. However, due to the graphical nature of wizards, they cannot be executed using MQL.

Wizards are composed of *frames*. Each frame contains one or more *widgets*. Each term is explained in detail below.

What is a Frame?

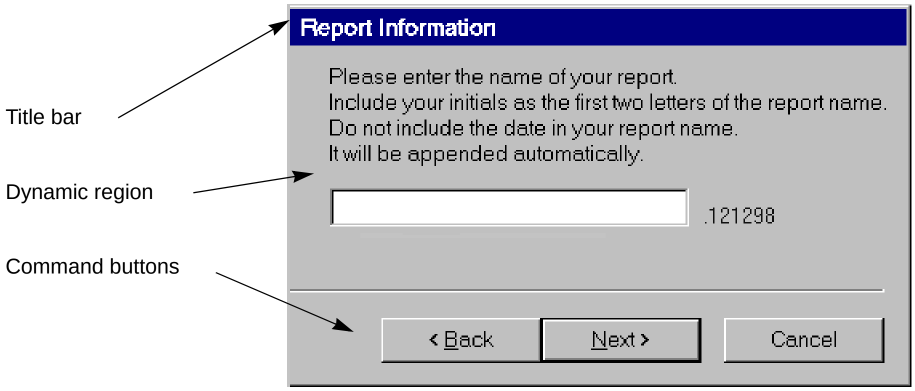
A *frame* is a dialog box that contains instructions and/or asks for information from the user. The information gathered in one frame can be checked for correctness before moving to the next frame and will often dictate the choices loaded into the next frame.

Frames are designed by the Business Administrator using a layout editor. The basic layout of a frame consists of a fixed title in the title bar at the top, a fixed set of three active command buttons along the bottom, and a dynamic region (sub-frame) in the middle. Only the dynamic region can change from one frame to the next.

In order to be effective, a wizard should gather information over a series of simple steps with each step responsible for a single piece of information. This allows for focused activity during each frame and eliminates the need for large complex dialogs.

The frame-specific dynamic region typically has a multi-line text box near the top that describes the step being performed. The rest of the region is set up to gather the information needed for the step to be completed. The region is called “dynamic” because choices provided in the region can be filled during run time.

The following shows a frame whose dynamic region contains a multiline textbox which explains the format for the name of a report. It also provides an editable text box where the user can type the report name.



Please enter the name of your report.
Include your initials as the first two letters of the report name.
Do not include the date in your report name.
It will be appended automatically.

.121298

For most frames, three command buttons are available: **Back**, **Next**, and **Cancel**. After providing information or making a selection, the user clicks the **Next** button to proceed to the next frame. Clicking the **Back** button returns to the previous frame, where changes can be made to the information supplied. Clicking the **Cancel** button exits the wizard. The first frame of any wizard has a deactivated (grayed-out) **Back** button. The final frame of the sequence has a **Finish** button in the place of the **Next** button. When a user selects the **Finish** button in the final frame, the task is carried out by executing the code associated with the Business wizard. A single optional status frame can be used to return feedback to the user about the task just performed (or provide the reason for the task not being performed).

What is a Widget?

Anything that is included in the dynamic region of a frame is called a *widget*. The designer of the wizard chooses how many and what types of widgets will be included in each frame.

A frame can include any number of widgets, but it is most effective to reduce the complexity of each frame.

There are ten different types of widgets. The following tables lists the widget types, which are explained in more detail in *Working With Widgets*.

Widget	Icon	Description
label	A	a non-editable text field
command button	▭	a button which, when clicked, executes a program
text box	abl	a text field which can be defined to be either editable or non-editable
image	🖼	a picture file containing a GIF image
list box	▤	a list of items from which a single or multiple selections can be made
combo box	▤	a combination of an editable text box and a list box—users can type their choice or select from the list
radio button	◉	a group of one or more mutually exclusive options, each with a circle beside it; only one can be selected
check box	☑	a group of one or more options, each with a check box beside it; multiple options can be selected
file text box	📄	a text field with an ellipsis button next to it, which launches the file chooser.
directory text box	📁	a text field with an ellipsis button next to it, which launches the directory chooser.

Runtime Program Environment

Business wizards require a mechanism for passing information between the Program objects that can be used within the wizard. The *Runtime Program Environment* (RPE) is provided to hold program environment variables. The RPE, which is a dynamic heap in memory that is used to hold name/value pairs of strings, makes it possible to pass parameters between internal programs and scripts.

For Business wizards, the RPE allows information to be passed between frames, between programs that control the individual widgets, and to the wizard program that processes the information gathered from the user. For example, after all data-gathering frames have been displayed, you may want to present a summary frame where the user can check for errors. Also, if the user clicks the **Back** button, you would want the frames to display just as they did when the user clicked the **Next** button, including the choices the user made.

Business Wizard Internal Programs

The basic functions of the Business wizard require no specific programming since they are handled internally by the system. These basic functions include displaying frames and widgets, including responding to the Next, Back and Cancel buttons.

Business wizard internal programs should not launch external applications, such as a word processing program. Attempts to do so will cause unexpected results, including crashing the wizard.

The Business Administrator customizes each Business wizard by writing programs that control the individual widgets and that execute a task based on the information collected from the user. The programs that can be used to customize the Business wizard include the following:

- **Prologue** – the program that executes just before a frame is displayed. See *Defining Prologues and Epilogues*.
- **Epilogue** – the program that executes when a frame is closed. See *Defining Prologues and Epilogues*.
- **Load Program** – the program that loads values into the widget field. See *Defining Load and Validate Programs*.
- **Validate Program** – the program that tests the validity of the widget's state. See *Defining Load and Validate Programs*.
- **Button Program** – the program that controls what happens when a command button located in the dynamic area of a frame is selected. See *Defining a Program*.
- **Wizard Code** – the program that controls the task that is executed when the user clicks the Finish button in the Business wizard. See *Adding Code*.

All of the programs listed above (except for the actual wizard code) must be defined as independent program objects before they can be used by a Business wizard. You can edit existing programs while defining a wizard but you cannot create a new program object while editing a wizard.

Creating a Business Wizard

Creating a Business wizard involves setting up basic wizard parameters, defining frames and widgets (the components of frames), and specifying the underlying program code.

Planning the Wizard

Before you start creating a Business wizard, you should plan what the wizard will accomplish and how you want it to look.

- Decide which task(s) the business wizard will perform, for example, checking in a file.
- Determine what information is needed to perform the task, for example, user name, department name, report name, etc.
- Based on the amount of information that needs to be collected, decide how many frames are needed and how each frame should look.

Basic Procedure

The following procedure describes the general steps involved in creating a business wizard. These steps are explained in detail in the sections that follow.

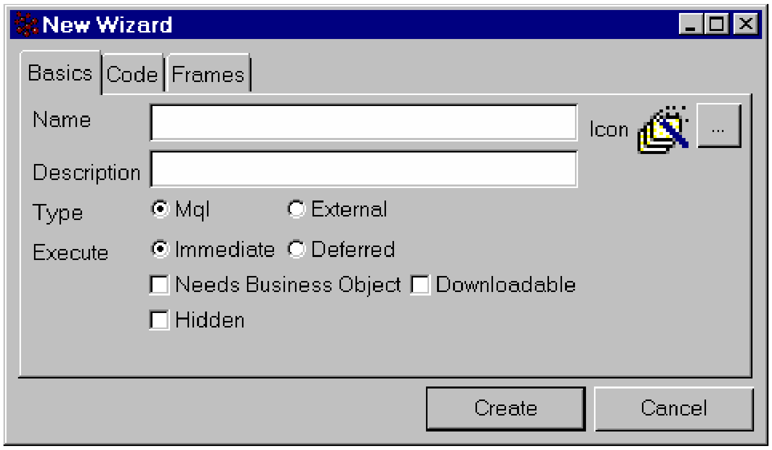
1. Define general information about the business wizard.
2. Establish the sequence of frames (much like defining lifecycle states in a Policy). See *Working With Frames*.
3. Define each frame, using the Frame layout dialog.
4. Position and define the widgets for each frame, associating them with optional programs that load and validate data. See *Working With Widgets*.
5. Type in the code for the wizard or copy it from another file. See *Adding Code*.

New Wizard Dialog Box

To create a business wizard

1. Click or select **Wizard** from the Object>New menu.

   The New Wizard dialog box opens, showing the **Basics** tab.



Name

Description

Type ● Mql ○ External

Execute ● Immediate ○ Deferred

□ Needs Business Object □ Downloadable

□ Hidden

2. Assign values to the Name, Description, and Type parameters, as appropriate.

Each parameter in the New Wizard dialog box and its possible values are discussed in the sections that follow.

Defining a Name

Enter a unique name for the Business wizard, using any combination of alphanumeric characters and spaces. The wizard name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

Assigning an Icon

Click ... next to the icon to display the Icon Chooser, where you can select an icon for the new Business wizard. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

The icon assigned to a Business wizard is also considered the ImageIcon of the wizard. When an object is viewed as either an icon or ImageIcon, the GIF file associated with it will be displayed.

Defining a Description

A description can provide general information to you and the reader about the function of the wizard. Since there may be subtle differences between wizards, you can use the description to point out the differences to the reader.

There is no limit to the number of characters you can include in the description. However, keep in mind that the description appears when the mouse pointer stops over the wizard in the Program chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Defining the Type

Specify the source of the code which will control the Business wizard. Select **MQL** or **External**.

An MQL program can be run from any machine with Live Collaboration installed; it is not necessary for MQL to be available on the machine.

Defining the Execute Option

Specify when the wizard should be executed. Select **Immediate** or **Deferred**.

Immediate execution means that the program runs within the current transaction, and therefore can influence the success or failure of the transaction, and that all the program's database updates are subject to the outcome of the transaction.

Deferred execution means that the program is cued up to begin execution only after the outer-most transaction is successfully committed. A deferred program will not execute at all if the outer transaction is aborted. A deferred program failure only affects the new isolated transaction in which it is run (the original transaction from which the program was launched will have already been successfully committed).

However, there are a number of cases where deferring execution of a program does not make sense. In these cases the system will execute the program immediately, rather than deferring it until the transaction is committed. Deferred execution is ignored in the following cases:

- Wizard frame prologue/epilogue
- Wizard widget load/validate
- Format edit/view/print

A program downloaded to the web client for local execution (see *Defining the Downloadable Option*) can be run only in a deferred mode. Therefore, selecting the **Downloadable** option automatically sets the Execute option to Deferred.

As of 3DEXPERIENCER2015x, the Web Navigator app is no longer supported.

Specifying the Need for a Business Object

You can specify that the wizard must function with a business object by checking the **Needs Business Object** check box. This selection assumes that you will be adding the Business wizard to a business Type as a method.

When a wizard is added as a method to a type, then that wizard needs a business object in order to execute even if Needs Business Object box was not enabled for that wizard.

For example, you would select this option if the wizard promotes a business object. If however, the wizard creates a business object, the wizard is independent of an existing object and this option would not apply.

When a user opens the Program Chooser, the programs displayed are associated with the type of the selected object. Live Collaboration runs a program that requires a business object with the selected object as the starting point. If the program does not require a business object, the selected object would not be affected.

Defining the Downloadable Option

If the wizard includes code for operations that are not supported on the Web product (for example, Tk dialogs or reads/writes to a local file) you can check the **Downloadable** box. If this is checked, this program is downloaded to the Web client for execution (as opposed to running on the server). For wizards not run on the Web product, this flag has no meaning.

As of 3DEXPERIENCER2015x, the Web Navigator app is no longer supported.

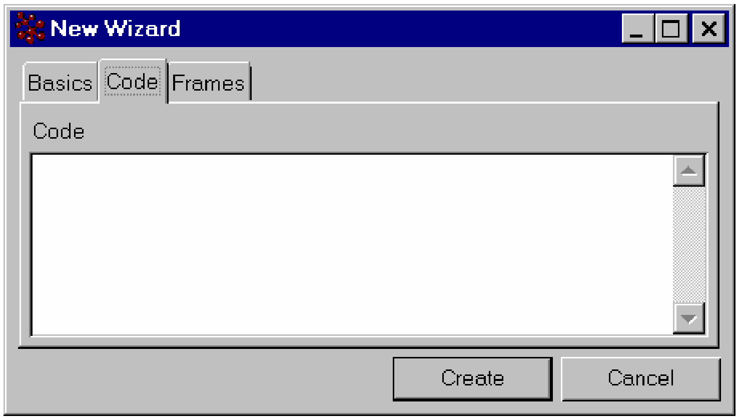
Defining the Hidden Option

You can mark the new wizard as “hidden” so it does not appear in the Program or Methods choosers in Live Collaboration, which simplifies the end-user interface. A wizard can contain a number of programs which are not intended to be executed as stand-alone programs (such as the load and validate programs for widgets) and users should not be able to view these program names in the Program chooser. All programs related to wizards other than the actual wizard program should be marked hidden. Users who are aware of a hidden program’s existence can enter its name manually where appropriate. Hidden objects are also accessible through MQL.

Adding Code

To add code for the Business wizard

- Click the **Code** tab in the New Wizard dialog box.



Include MQL/tcl program commands in this text box if you selected the **MQL** option on the **Basics** tab. The code included in this text box provides the instructions to act on the information collected from the user during the execution of the frames belonging to the Business wizard.

If you prefer, you can write the code in another editor and copy and paste it into this text box. Generally, the code would be written after all frames and widgets have been defined since it generally will act upon information specific to these objects

If you selected the **External** option, include in this text box the command necessary to start the external program. Include path information, if necessary. For example:

```
ENOVIA_INSTALL\programs\buswiz3.exe
```

See *Programming Business Wizards* for more information.

Working With Frames

A Business wizard is composed of one or more *frames*. Each frame is a window that requests and stores information in order to complete a task. You design frames using a layout editor. You can include any amount of information in a frame, but in order to be effective, each frame should be responsible for gathering a single piece of information, for example, a department name or the name of an object in the database.

The basic layout of each frame contains three components. The title bar and command buttons are handled internally and require no programming to make them visible at runtime. Note that they do not appear on the frame layout at design time. Only the dynamic region of the frame is included in the design layout.

- **Title bar** — The title bar at the top of each frame shows the name of the Business wizard.
- **Command buttons** — Each frame has three command buttons at the bottom of the frame: **Back**, **Next**, and **Cancel**. At runtime, clicking the **Next** button allows the user to move to the next frame in the sequence. The **Back** button displays the previous frame. The **Cancel** button exits the Business wizard.

On the first frame of the wizard, the **Back** *button is grayed out. On the last frame of the wizard, the* **Next** *button is replaced by a* **Finish** *button.*

- **Dynamic region** — This is the section of the frame you design. Each frame's dynamic region is different. It can contain graphics, choices, and editable or non-editable text. These component building blocks of the dynamic region are called *widgets*. Widgets are explained in detail in the section *Working With Widgets*.

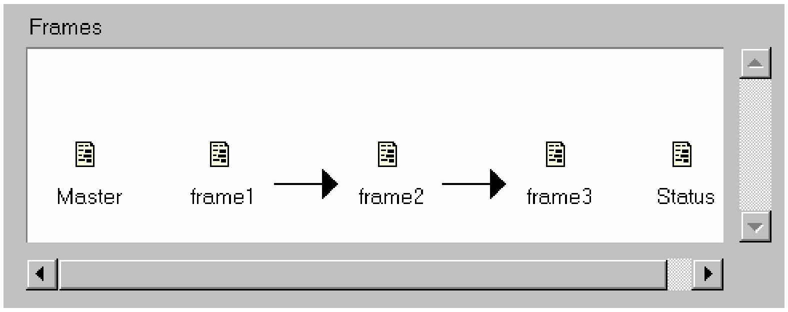
The final frame of the sequence should explain what will happen when the user clicks the **Finish** *button. It could give a summary of the information the user has provided and ask if any of the information needs to be changed. This allows the user a chance to return to previously-displayed frames if something minor needs correcting, or cancel completely out of the wizard.*

You can define as many individual frames as your application requires. As you create frames in a wizard, they are connected by arrows showing their sequential order. You can add or remove frames anywhere in the sequence as needed.

Special Frames

In addition to the basic sequential frame, Business wizards have two types of special frames: *master frames* and *status frames*. These frames are defined in the same way as other frames in the sequence, but they have specialized tasks. See the sections *Editing a Frame* and *Working With Widgets* for information on how to define a frame.

The master frame and status frame are not part of the frame sequence. They are displayed in the **Frames** section of the New Wizard dialog box as separate frames, not connected by arrows to the rest of the frame sequence.



Master Frame

One frame is defined by default in the Frames area of the New Wizard dialog box. The *master frame* serves as a template for other frames in the wizard. It makes it easy for all frames to have the same basic “look and feel.”

The master frame is not part of the frame sequence, but influences the look of each frame in the sequence. All characteristics of the master frame are inherited by every other frame in the wizard. You might, for example, want to have a logo on every frame, or you might want to define a color scheme for all frames.

When new frames are added to a wizard, they are given the size and color of the master frame (as currently defined). These default attributes can be overridden, as needed, on a frame-by-frame basis. Keep in mind that only new frames take on the shape and color of the master frame. Frames that were previously added to the Business wizard will *not* automatically change if the dimensions or color of the master frame is modified. Therefore, it is best to get the size and color of your master frame correct *before* adding frames to make up the frame sequence.

Any widget included in the master frame is automatically included in each frame of the sequence. This serves two purposes: first, the widget is guaranteed to appear in the exact same position in each frame; and second, the widget is stored only once. This frees up space in the database. Also, since widgets are loaded at run time, any change to the master widgets is reflected in all frames of the sequence. Widgets, therefore, can be modified at any time during the construction of the wizard.

Status Frame

Each wizard can have an optional *status frame*, which is generated after the user has clicked the Finish button and the Business wizard code has been executed. A special option, **Status Frame**, is available on the Format tab of the Edit Frame dialog box on the last frame of the sequence. Set this option to **True** if you want this frame to be the status frame. When a status frame is present, it appears at the end of the frame sequence and is not joined by an arrow.

The status frame returns feedback to the user about the task just performed or provides the reason why the task was not performed. The status frame contains a single **Close** button in place of the three regular frame buttons (Back, Next, Cancel). The dynamic region would typically contain only a multi-line text box, though it could also contain graphic or label widgets.

Status frame processing includes executing the prologue and the load programs, but no input from the user is accepted or processed. The status frame programs should not perform any database operations.

If the **Status Frame** option is set to **False** (this is the default), no status frame is generated at the conclusion of the Business wizard.

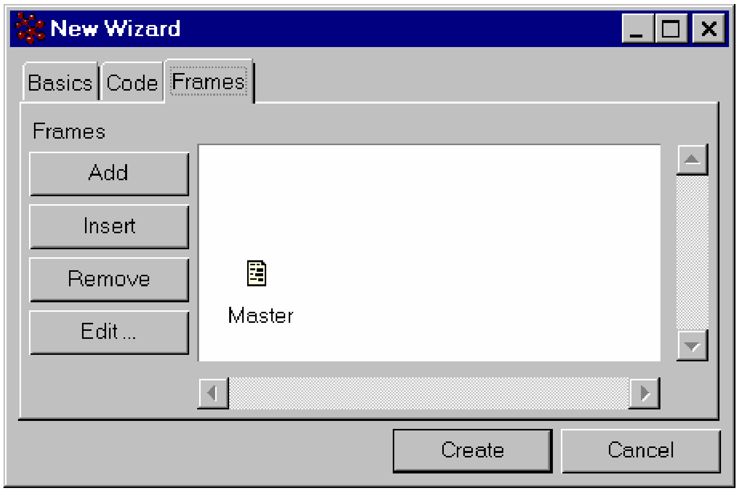
Adding or Inserting Frames

You can add a frame at the end of the sequence, or add/insert a frame at a specific point in the sequence.

Remember, if you want to have a standard color and size for all frames, you should edit the Master Frame before adding new frames.

To add a frame

1. Click the Frames tab in the New Wizard dialog box.



2. Select an existing frame. If you want to add a frame to the end of the sequence, you can skip this step.
3. Click **Add** in the **Frames** area of the New Wizard dialog box. The new frame is automatically named:

- If no frame is selected, the new frame is added to the end of the sequence and named *framex* where *x* is the next number in the sequence (frame1, frame2, etc.).
- If a frame is selected, the new frame is added *after* the selected frame. It is named *post* plus the current frame name. For example, if the frame *getPassword* is selected, the new frame is named *postgetPassword*.

To insert a frame

1. Select an existing frame. If you want to insert a frame at the beginning of the sequence, you can skip this step.
2. Click **Insert** in the **Frames** area of the New Wizard dialog box. The new frame is automatically named:

- If no frame is selected, the new frame is added to the beginning of the sequence, but after the Master frame. The new frame is named *framex* where *x* is the next number in the sequence (frame3, frame4, etc.).

- If a frame is selected, the new frame is added *before* the selected frame. It is named *pre* plus the current frame name. For example, if the frame *getPassword* is selected, the new frame is named *pregetPassword*.

To remove a frame

1. In the New Wizard dialog box, click the **Frames** tab.
2. Select the frame that you want to remove.
3. Click **Remove**.

Editing a Frame

Once you have added or inserted frames in the **Frames** area of the New Wizard dialog box, you need to design the layout, content, and associated programs for each frame.

To edit a frame

1. From the **Frames** tab in the New Wizard dialog box, double-click on the frame.

   Or

   Select a frame and click **Edit**.

   The Edit Frame dialog box opens, showing the **Basics** tab:
2. Assign values to the parameters, as appropriate. Each parameter and its possible values are discussed in the sections that follow.
3. When all values have been assigned and all necessary widgets added, click **Edit** to create the frame.

Defining a Frame Name

Enter a name for the frame you are creating. You can keep the name automatically assigned to the frame by the system, or create a different one. It is generally more useful to assign a name that has meaning to both you and the user. The frame name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

The name of a wizard frame must be unique only among the frames belonging to the owning Business wizard. The following are some examples of names you might use

Department Name
Get Codeword
Get Filename
Availability Choices

Assigning a Frame Icon

Click next to the icon to assign an icon to the new frame. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Frame Description

A description provides general information about the function of the frame. Since there may be subtle differences between frames, you can use the description to point out the differences to the reader.

Defining Prologues and Epilogues

A prologue program executes just before a frame is displayed and can be used to prepare for the loading of the widgets. An epilogue program executes when a frame is closed and can be used to perform clean up activity for the frame.

In the **Input** field to the right of the **Prologue** or **Epilogue** field, you can define arguments to be passed into the program. The arguments can then be referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

- Environment variable “0” always holds the program name and is set automatically by the system.
- Arguments from the **Input** field are set in environment variables “1”, “2”, . . . etc.

See *Programming Business Wizards* for additional information on program environment variables.

Prologue

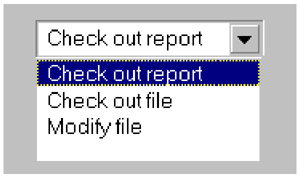
The frames’s *prologue* is a program which executes immediately before a frame is displayed and before any other action is taken. Prologue programs can be used to influence the loading of widgets. They can also be used to skip frames. Each frame within a Business wizard can have its own prologue.

To choose a prologue program, click to display the Program Chooser or type the name of the program to run.

To edit the prologue program named in the input field, click to display the Edit Program dialog box.

The prologue program can be used to prepare for the loading of the widgets. This might include setting values in the RPE, and even redefining widget load programs. This allows widget load functions to be generic, with specific behavior being controlled at run time by the owning frame.

For example, you may have a combo box that lists the following options:



But for most users, only the “check out” options should be available, not the “modify” option. Use the prologue to set RPE variables defining which options should be included in the combo box, based on context or on some other information gathered from a previous frame.

The prologue function can also cause the frame to be skipped. Whenever a prologue program returns a non-zero value, the frame is skipped.

See *Sample Programs* for an example of a prologue program.

Epilogue

The frame's *epilogue* is a program which is executed whenever a frame is closed. The epilogue program can be used to undo what the prologue program did, and perform any other cleanup activity. Since data is passed between frames using RPE, you can use the epilogue program to clean up widget variables before moving on to the next frame.

If an epilogue program returns non-zero after the user presses the **Next** button, the current frame is redisplayed. Thus, even if all the widget validate programs return zero, the epilogue program can still keep the next frame from appearing. Although the epilogue program runs when the **Back** button is pressed, the exit code is not checked.

When the epilogue program for the last frame in the frame sequence has been executed, programmatic control of the wizard is then handled by the wizard code, defined on the Code tab of the Business wizard. See *Adding Code* for additional information.

To choose an epilogue program, click to display the Program Chooser or type the name of the program to run.

To edit the epilogue program named in the input field, click to display the Edit Program dialog box.

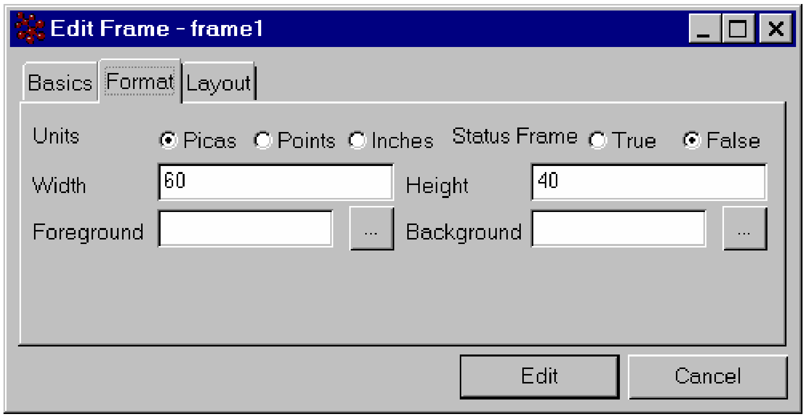
See *Sample Programs* for an example of an epilogue program.

Defining the Frame's Format

The frame's format information includes the units of measurement and the size and color of the frame.

To define the frame's format

1. Click the **Formats** tab in the Edit Frame dialog box.



Edit Frame - frame1

Basics | Format | Layout

Units ● Picas ○ Points ○ Inches Status Frame ○ True ● False

Width 60 Height 40

Foreground ... Background ...

2. Assign values to the parameters, as appropriate. Each parameter and its possible values are discussed in the sections that follow.

Defining Units

Indicate the units of measurement to be used to define the size of the frame. Choose **Picas** (fixed size characters that render 80 by 66 on an 8-1/2 x 11 sheet), **Points** (graphical units of 72 points per inch), or **Inches**.

When you are viewing a frame definition, as described in *Viewing an Administrative Object* in Chapter 1, you can select a different unit type (Picas, Points, or Inches) and the measurements change accordingly.

If you choose a different unit of measurement, click the **Layout** button on the Layout tab to immediately see the new dimensions take effect.

Defining Width and Height

Define the page dimensions of the frame. A frame can be any size that your monitor can handle. Enter values for the **Width** and **Height** of the page. The values are measured in the units selected.

When defining the frame size, keep in mind the monitor sizes of the users who will be running the wizard. For example, a wizard frame does not display the same amount of information on a screen with a resolution of 640x480 as it does on a screen with a resolution of 1280x1024. Use the lowest common denominator of screen resolution to design wizards, and the master frame should be fit to size.

If you choose a different width or height, click the **Layout** tab to immediately see the new dimensions take effect.

Defining Color

To define the frame's colors, click ... to display the Color Chooser or type the name of a color in the **Foreground** and the **Background** text boxes. Choose foreground and background colors that complement each other. Colors that offer high contrast together work best.

If you choose different colors, click the **Layout** tab to immediately see the new colors.

Creating the Frame

After you have supplied information for the frame and placed all appropriate widgets on the frame grid, click **Edit** in the Edit Frame dialog box to create the frame. See the next section, *Working With Widgets* for information about how to add widgets to the frame.

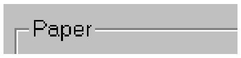
Working With Widgets

After entering the appropriate information in the Edit Frame dialog box, you are ready to create widgets on the frame. The term *widget* refers to any component of the frame.

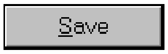
The following shows widget types and examples of what the widgets might look like:

Widget types include the following:

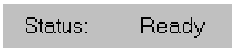
- A label — a non-editable text field.



- command button — a button which launches a program object.

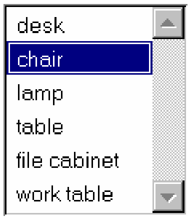


- text box — a text field which can be defined to be either editable or non-editable.

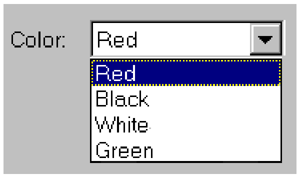


- image — a picture file containing a GIF image.

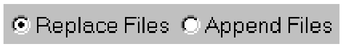
- list box — a box containing a list of items from which multiple selections can be made.



- combo box — combines features of an editable text box and a list box. Users can enter text or select a single value from the displayed list.

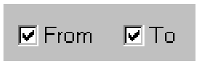


- radio button — a group of one or more options, each with a circle next to it. A blank circle indicates that it is not selected (or “off”); a filled in circle indicates that it is selected (or “on”). Radio buttons are used for mutually exclusive choices, that is, only one option can be selected. For example:



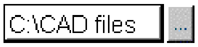
● Replace Files ○ Append Files

- ☑ check box — a group of one or more options, each with a check box next to it. A blank check box indicates that it is not selected (or “off”); a box with a check in it indicates that it is selected (or “on”). Multiple options can be selected. For example:



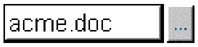
☑ From ☑ To

- 📁 directory text box — an editable text box with an ellipsis button next to it, which launches the directory chooser. For example:



C:\CAD files ...

- 🗎 file text box — an editable text box with an ellipsis button next to it, which launches the file chooser. For example:



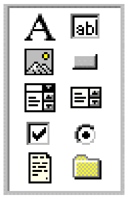
acme.doc ...

Creating Widgets

Widgets are added to a frame using a drag-and-drop procedure, that is, you click on a widget and drag it onto the layout grid which represents the dynamic region of the frame. You then double-click on the widget to display a dialog box where you can specify options for the widget.

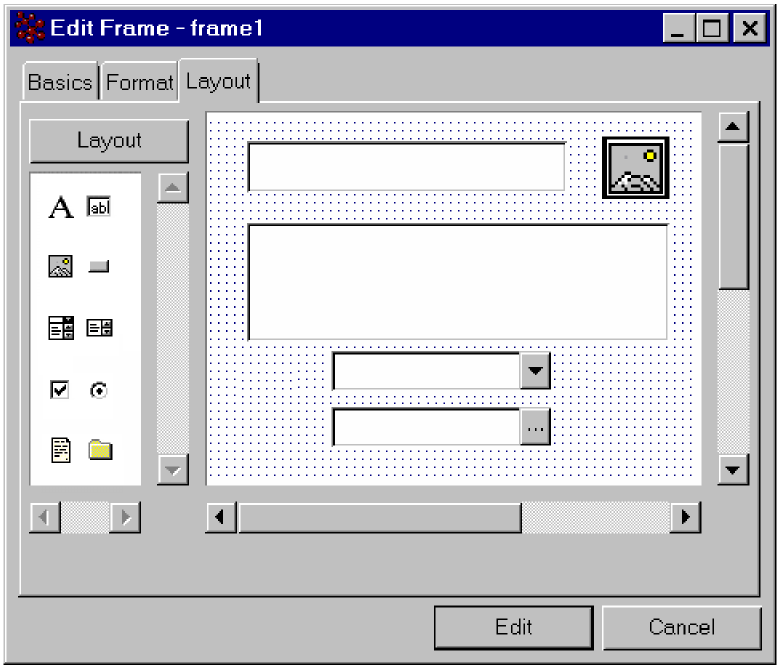
To add widgets to a frame

1. With a left-mouse click, select a widget type from the widget toolbox which is located to the left of the frame grid. As you position your mouse pointer over a widget, the name of the widget type is displayed.



2. Drag the widget from the box and drop it on the appropriate location on the frame grid.

   The widget is displayed in a white area on the grid or as an icon if the widget type is graphical. The following shows a label and an image at the top of the frame, a text box in the middle, and a combo box and directory text box at the bottom.



3. Double-click the widget area on the grid to edit the widget.

The Edit Widget dialog box specific to the type of widget is displayed.

4. Enter information about the widget on this dialog box. Detailed descriptions of these fields are included in the sections that follow.

5. After entering all field information, click **Edit** to accept the changes and return to the Edit Frame dialog box.

To change the position of a widget on the grid

- Click in the middle of the white field area on the grid and drag the widget to a new location.

  Or

  Change the X Y coordinate values on the Format tab of the Edit Widget dialog.

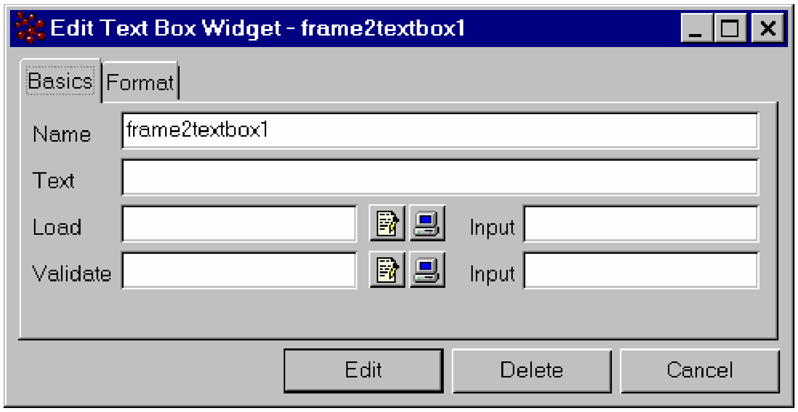
To change the size of a widget on the grid

- Click on the desired edge of the white field area on the grid and drag the edge to enlarge or reduce its size.

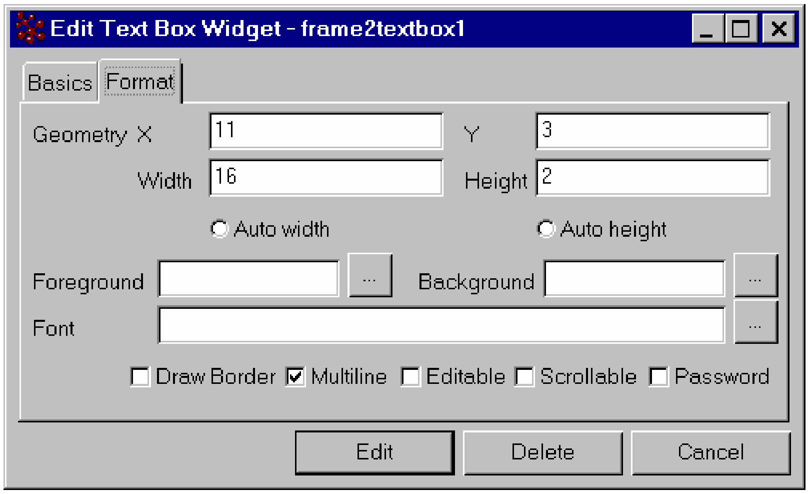
  Or

  Change the Width and Height values on the Format tab of the Edit Widget dialog.

Options on the Edit Widget dialog boxes vary depending on the type of widget. This shows the Edit Text Box Widget dialogs. The options are described in the following sections.



Name: frame2textbox1



Geometry X: 11 Y: 3

Width: 16 Height: 2

○ Auto width ○ Auto height

☐ Draw Border ☑ Multiline ☐ Editable ☐ Scrollable ☐ Password

Defining a Widget Name

Define a unique name for this widget. Widget names can be any length of alphanumeric characters; spaces can be included.

Since widgets are named so that they can be identified by the system, a unique name is automatically created by the system for each widget. This name is displayed by default in the **Name** field in the Edit Widget dialog box. You can choose to use this name, or replace it.

If you want to include the widget on multiple frames within the Business wizard and retain the value within each frame, you should replace the system-supplied name with a name of your own choosing. Widgets that share the same name within a wizard will share the same value.

The text displayed on widgets can support multiple languages, as long as the text is defined in the translation file. For example, if an alternate language matrix.txt file is imported and a button widget is named “Cancel” or “View” (or any other words set in “matrix.vr”), its display changes to the entry in the language file.

See *Programming Business Wizards* for additional information on widget names.

Defining Default Text

Specify the default text that will appear when the frame is loaded. This parameter is optional, since the default text can be defined in the Load program.

Text is an option for the text box, file text box, and directory text box widgets.

For an editable text box, you could include some informational text, such as “Type your Name here.” For a non-editable text box, you could include the information to be displayed in the box, and you would not need to define a load program.

The value of the widget variable is used as the default text in the widget. See *Programming Business Wizards* for information about how program environment variables are loaded.

Defining a Label

A label consists of any printable string of characters. **Label** is an option only on the Edit Label Widget dialog box.

Defining a Pixmap

Type the name of an image file to be displayed in the frame or click ... to choose an image from the Icon Chooser. You must select a GIF format file.

Pixmap is an option only on the Edit Image Widget dialog box.

Defining a Program

Type the name of the button program, or click 🖥 to choose a program name from the Program Chooser.

To edit the button program named in the input field, click 📝 to display the Edit Program dialog box.

Program is an option only on the Edit Command Button Widget dialog box. This specifies the name of the program that will execute when the user clicks the button.

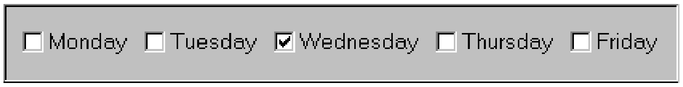
If a command button program returns non-zero, the frame is refreshed. This means that all widgets in the frame will have their load program run. This allows a command button program to modify variables in the RPE and then force the widgets in the current frame to reload themselves.

Choices and Selected

Several widgets need two pieces of information: a list of choices and one or more selections. For example, you might have a check box group that lists the days of the week and you want “Wednesday” to be selected by default. In the **Choices** field, type the days of the week, separated by spaces:

```
Monday Tuesday Wednesday Thursday Friday
```

In the **Selected** field, type `Wednesday`. The output would look similar to the following:



☐ Monday ☐ Tuesday ☑ Wednesday ☐ Thursday ☐ Friday

Choices and **Selected** are options for the list box, combo box, radio button and check box widgets.

For the list box and check box, additional selected options can be included, separated by spaces. If any choices include white space, use quotes. For the combo box and radio button widgets, only one selected option can be included in the **Selected** field.

These fields are optional. The choices and selected options can instead be handled in the load program for the widget.

Defining Load and Validate Programs

Load programs allow you to load values into a widget and set its state. Validate programs check if the values added by the user (to an input field, for example) are within an acceptable range of parameters.

In the **Input** field to the right of the **Load** or **Validate** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

- Environment variable “0” always holds the program name and is set automatically by the system.
- Environment variable “1” always holds the widget name and is also set automatically by the system.
- Arguments from the Input field are set in environment variables “2”, “3”, . . . etc.

See *Sample Programs* for sample load and validate programs.

Load Program

Type the name of the load program, or click 🖥 to choose a program name from the Program Chooser.

To edit the load program, click 📝 to display the Edit Program dialog box.

Load is an option for all widgets except for label and image.

The load program, if defined, contains code to load values into the widget field. The load program can also be used to define alternate values for the widget depending on the context, information gathered in a prior frame, etc.

Validate Program

Type the name of the validate program, or click 🖥 to choose a program name from the Program Chooser.

To edit the validate program, click 📝 to display the Edit Program dialog box.

Validate is an option for all widgets except for label and image.

The validate program, if defined, is used to test the validity of the widget’s state. By returning a non-zero value, the validate program causes the system to redisplay the frame and place focus on the offending widget. It is good style to have the validate program additionally display a message box that explains the error.

Validate programs are normally used for text box widgets that have the **Editable** option specified, and also for combo box widgets.

Note that the validate program does not execute when the user clicks the **Back** *button.*

Defining an Observer Program

An observer program can be assigned to a wizard list box, check box, or radio button. The observer program is immediately executed when the selection(s) in the corresponding widget change, and it indicates (via RPE) which widgets on the frame need to be reloaded and redrawn.

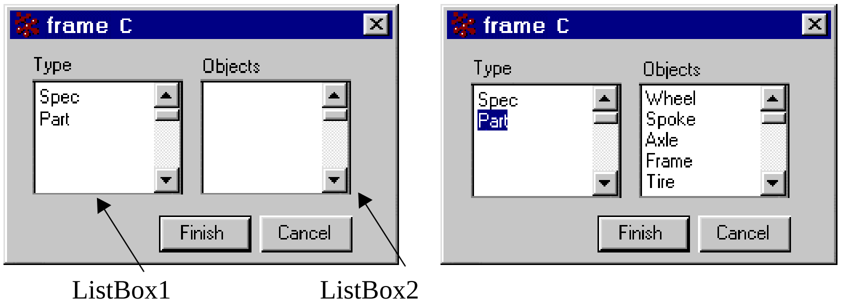
Type the name of the observer program, or click to choose a program name from the Program Chooser.

To edit the observer program, click to display the Edit Program dialog box.

In the **Input** field to the right of the **Observer** field, you can define arguments to be passed into the program. The arguments are referenced by variables within the program. Variables 0, 1, 2...etc. are reserved by the system for passing in arguments.

- Environment variable “0” always holds the program name and is set automatically by the system.
- Environment variable “1” always holds the widget name and is also set automatically by the system.
- Arguments from the Input field are set in environment variables “2”, “3”,. . . etc.

For example, the selection of Type “Part” from the ListBox1 calls an observer program that populates the Business Objects in ListBox2, as illustrated below.



The observer program has all the side effects of clicking a button within a wizard; that is, it is executed immediately and if it exits with a return code of 1, each of the widgets in the current frame has its values (re)read and (re)populated by the RPE variables associated with the frame. The INVOCATION macro is populated with the value “observer.” If the widget was a multi-selection list box, then `mql get env <widget name>` returns a space-delimited string containing current values.

Performance is an important consideration. When calling an observer program, every widget on a frame is examined to get the current value of that widget. For a frame with many widgets, this could be time consuming. Also, since the observer program is actually run on the server, there is lag time associated with the round trip from client to server and back. The developer of an observer program needs to evaluate whether the time it takes for the observer program to complete is acceptable for a user to wait.

Widget Geometry

You can define the size and placement of the widget in the frame.

- **X and Y —** Type X and Y values to specify the field location on the frame. The X (horizontal) and Y (vertical) coordinates identify the field’s starting point—where the first character of the field value is displayed. 0,0 is the upper left hand corner.

X and Y values can also be changed dynamically in the Edit Frame dialog box by clicking and dragging the widget to the desired location on the frame grid. Changes are reflected in the **X** *and* **Y** *fields in the Edit Widget dialog box.*

- **Width and Height —** Type values to specify the widget size. Widgets use the same units (picas, points or inches) that are defined for the frame. Enter a width value for the horizontal size of the widget. Enter a height value for the vertical size.

Width and height can also be changed dynamically in the Edit Frame dialog box by clicking and dragging the borders of the widget to the width and height desired. Changes are reflected in the **Width** *and* **Height** *fields in the Edit Widget dialog box.*

- **Auto Width and Height —** Select the Auto Height and Auto Width radio buttons if you want the program to select the appropriate height and width based on the contents of the widget.

Widget Characteristics

You can specify the font and colors for the widget, and also how the field is displayed to the user.

- **Foreground and Background —** Specify the color of the foreground (printed) information or background for the field.

  Type the name of a defined color or click ... to select from a list of colors.

Colors that can be displayed are based on the computer's system color settings.

- **Font —** Although it is possible to change the fonts used in wizards, it is best to let a wizard use the default font and colors which have been set up by the end user. If it is absolutely necessary to define specific fonts and colors for different widgets or frames, good style calls for minimizing this need.

  To change the font, specify the font in which the text of a widget field appears.

  Type the name of a defined font or click ... to select from a list of fonts. This option is available for all widget types except for the image widget.

Fonts available are based on the computer's defined system fonts.

- **Draw Border —** Select this option to include a border around the field. This option is available for the text box, image, check box, option button, file text box and directory text box widget types.
- **Multiline —** When this option is selected, text is displayed on more than one line, as necessary. This option is available for the text box widget type.
- **Multiselect —** Select this option to allow more than one choice to be selected in a list box. This option is available only for the list box widget type.
- **Editable —** Select this option to allow the user to change the value while viewing the frame. This option is available for the text box, file text box, and directory text box widget types.

  If you select Editable when editing either the file or directory text box widget, the wizard user can either use the ellipsis button to browse for the file or directory, or type in the path and/or filename. If you don't select Editable, they must use the ellipsis button.

- **Scrollable —** When this option is selected, scroll bars are displayed so that the user can scroll text up and down. This option is available only for the text box widget type. (This option is not included for the list box widget because it is an inherent quality of a list box.)
- **Password —** When this option is selected, all text typed into the field is masked with the asterisk character. This option is available only for the text box widget type.

Removing a Widget from a Frame

To remove a widget from the frame, double-click on the layout grid to select the widget and then click the **Delete** button at the bottom of the Edit Widget dialog box.

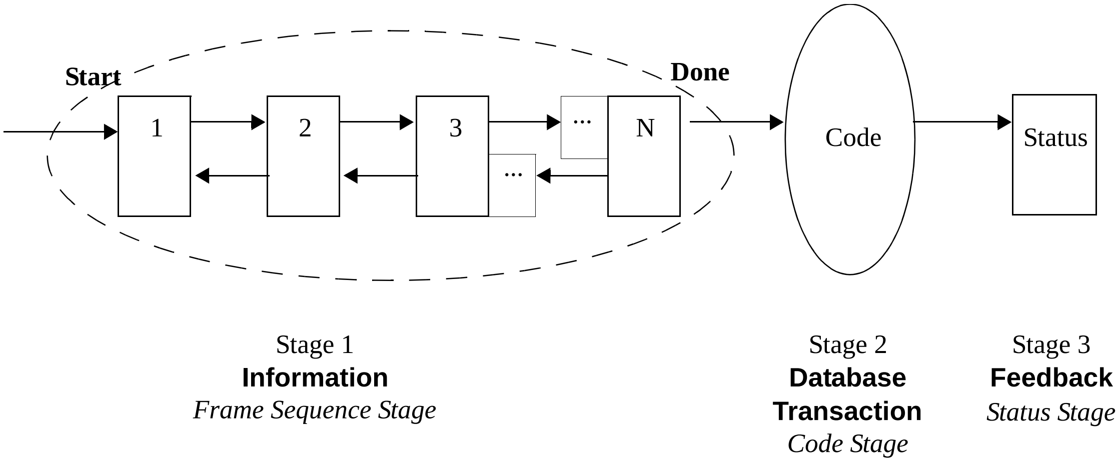
Programming Business Wizards

In creating a Business wizard, you will need to write at least one program, which will perform the database transaction(s) for which the wizard is created. You can also write programs to control the loading of frames and widgets and to check the validity of widget values.

Before you begin to create a Business wizard, you should be familiar with the information contained in *Overview of Programs* in Chapter 11, including information about the Runtime Program Environment (RPE).

As you plan the project, keep in mind that a wizard has three stages:

- **Stage 1** — Information. This stage consists of a series of frames, which gather information from the user. The information is stored in the RPE, which is used to pass data between frames. **No database updates should be performed during this stage.** This is important so that the **Cancel** function can work properly.
- **Stage 2** — Database transaction. The wizard code, defined in the New Wizard dialog box Code tab, is executed. This stage uses the information from Stage 1 to perform all necessary database transactions. Results can be placed in the RPE for use by the optional Status Frame.
- **Stage 3** — Feedback. A single status frame is displayed to inform the user of the status of the transaction performed by the wizard. This stage is optional. It is important to note that this stage follows stage 2 and is not part of the sequence of frames in Stage 1.



Strategy for Creating Business Wizards

The following is one strategy that can be used in creating a Business wizard. The order of the steps can, of course, be changed according to your own programming style.

1. **Plan the project.** Decide what questions you want to ask the user and what format they will use to supply answers. Will they type in their answers, or choose from a list or group? Input fields provide the most flexibility, but it is easier to control the user’s choices and prevent errors by having them make a selection from a fixed number of items.

2. **Design the wizard.** Decide the number of frames and the layout of the widgets in each frame. It is most effective to have each frame request a single piece of information.

   The frame sequence should only be used for the task of gathering information from the user, deferring any database work until the internal code of the wizard is executed.

3. **Design the master frame.** Master frames allow you to easily create a polished look for the Business wizard by making each frame the same size with the same color scheme. You can also add widgets that you want to appear in each frame, for example a logo or a horizontal rule. These default attributes can be overridden, as needed, on a frame-by-frame basis.

4. **Create the wizard**, positioning widgets on the frames. As you add frames, they inherit the characteristics of the master frame. If, however, you subsequently make changes to the colors or the dimensions of the master frame, these changes will *not* be reflected in any frames already added. Therefore, be sure to set up the master frame before adding any new frames.

5. **Write code to load and validate widgets**, if the widgets require load or validate programs.

6. **Write code for prologue and epilogue programs**, if frames require prologue or epilogue programs.

7. **Add program names** (for load, validate, prologue, epilogue) on the Edit Widget and Edit Frame dialog boxes of the Business wizard you created.

8. **Write code to execute the task** based on the information collected in the wizard. Include the code on the New Wizard dialog box of the Business wizard.

Program Environment Variables

Because business wizards are made up of a number of independent programs that may need to share data, there must be a way to pass information between the programs. The method used is the Runtime Program Environment (RPE), a dynamic heap in memory that is used to hold name/value pairs of strings.

In using the RPE to pass data between program objects, careful attention must be paid to the naming of program environment variables and to the cleaning up of these variables. All data is passed in the form of strings. Here are some guidelines:

- The program name is automatically associated with a variable named “0”.
- The widget name is automatically associated with a variable named “1”.
- Program input parameters are named “1”, “2”, etc. except within a load or validate program for a widget. Since the variable “1” is reserved for the widget name, program input parameters are named starting with the number “2”.
- The calling program can manually place the expected local variables into the RPE (using the naming scheme above) before executing the called program, or simply add them to the end of the execute command; the system will automatically place them in the RPE with the proper names.
- Programs should return their data in a variable with the name identified with the 1st input parameter.
- Lists should be returned using a Tcl form (space separated, with curly braces used to surround items that contain spaces or special characters).

- Since the RPE is shared memory, any variable can be overwritten by a variable of the same name. It is good practice to always capture wizard variables immediately into custom local variables to be assured that the values will be available where necessary within the wizard. RPE variables can be saved as local variables (in Tcl: `set localUser [mql get env USER]`) or saved back into the RPE under a different name that won't be overwritten (for example: `programName.USER` where programName is the name of the program).

  Any Live Collaboration program or trigger that runs at any time before the wizard completes has the potential to alter RPE variables. For example, a user starts a wizard, then checks the attributes of another object in the database. Depending on how the object is configured, a program or trigger may fire automatically without the user's knowledge that could overwrite existing RPE variables.

  For this reason, saving the Type, Name, Revision, and Objectid into wizard-specific RPE variables should always be done in the prologue of the first frame. This is the only way to preserve this information. A properly written wizard/program should *always* fetch out of the RPE all variables needed before proceeding.

How the System Displays Frames and Widgets

With the previous naming conventions in mind, here is how the frame and widget functions work:

1. When a frame is displayed, its prologue function is first executed. The prologue should establish any parameters in the RPE that will be used by the load functions of the widgets belonging to the frame. This allows widget load functions to be generic, with specific behavior being controlled at run time by the owning frame. The prologue function can also cause the frame to be skipped by returning a non-zero value.
2. Each widget belonging to the frame will then be constructed and set using the following steps:

   a) If an RPE variable exists with the widget's name, the system uses its value. The variable may already exist if a previous frame has a widget of the same name, or if the frame's prologue program created the variable.

   This ensures that any redisplay of a frame retains the previously-set value of the widget. It also allows you to include in a summary frame all items selected by the user.

   b) Assuming no RPE variable is found, the system next checks to see if the widget has a load program. If so, the load program is executed, and again a search is made for an RPE variable with the widget's name. (Each load program is given the widget's name in argument 1.)

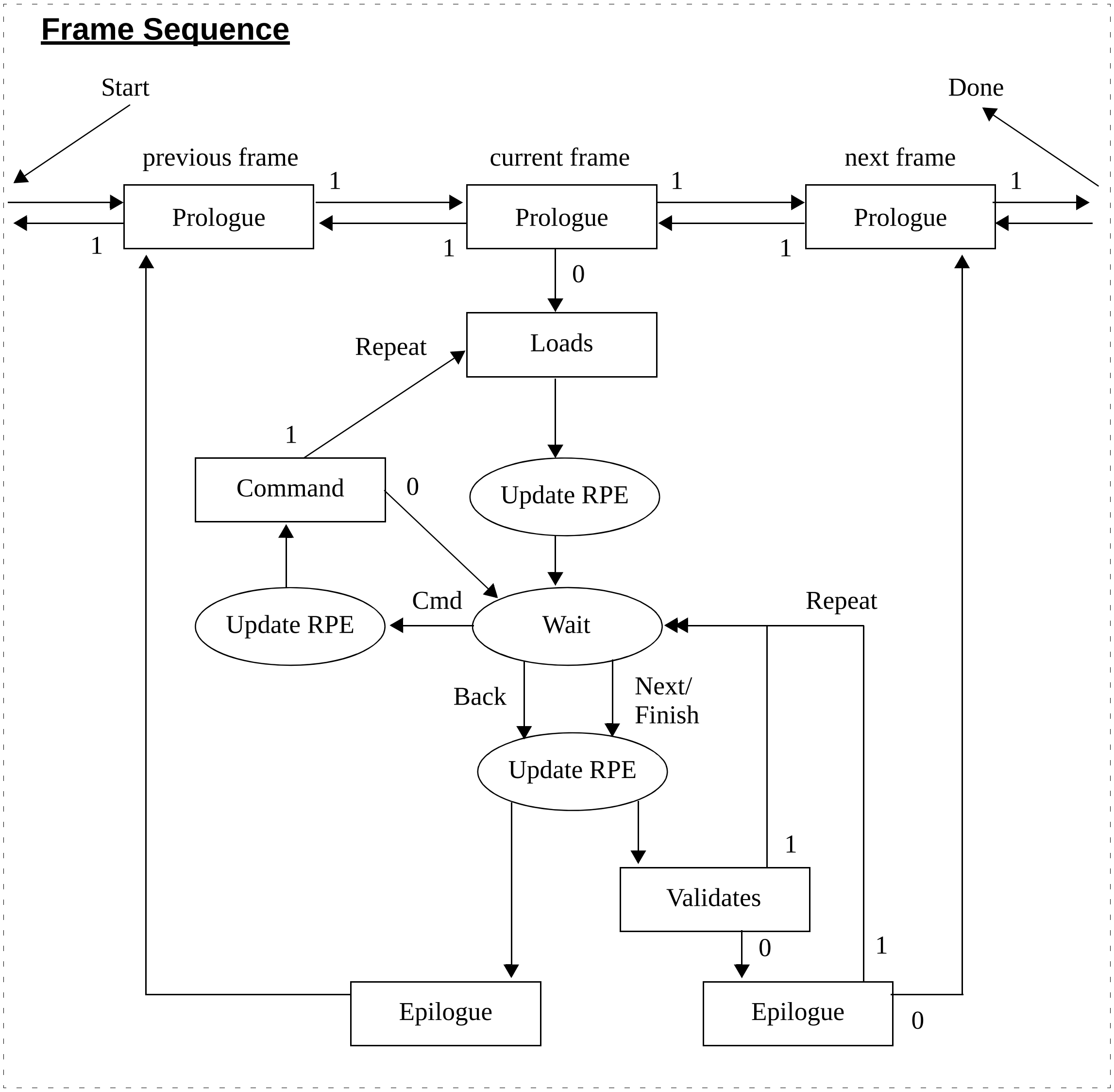
   c) Assuming no RPE variable is found, the system uses the widget's default value found in the value fields on the Edit Widget dialog box.
3. All widgets, except Label and Graphic, have an RPE variable that has the widget's name and has a value equal to its current state. For example, a widget named `answer` is a radio group made up of the choices `Yes` and `No`. This is represented in the RPE with a variable named `answer`; its value is either `Yes` or `No` depending on the widget's state. As the state of the widget changes, the value of the widget variable changes.

4. If the **Back** button is pressed, the current frame's epilogue function is called (performing any necessary cleanup), and steps 1 through 3 are repeated for the previous frame.

5. If the **Cancel** button is pressed, the Business wizard immediately ends.

6. If the **Next** or **Finish** button is pressed, each widget (except Labels and Graphics) belonging to the frame executes its validate function. After each save function is executed, the exit code is checked and if it is non-zero, the frame remains displayed and focus is placed on the offending widget. This should happen only with an editable text box or combo box widget since all other widgets should provide the user only with legal choices. The current frame's epilogue function is called, and if there is a next frame, steps 1 through 3 are repeated for that frame.

7. If the **Finish** button is pressed, then the body of the Business wizard code is executed. If a status frame is defined, it is displayed after the code is executed and only steps 1 and 2 are performed (the user will need to press the **Close** button to remove the frame from the screen –essentially step 5).



For each widget type there is a defined format for the widget variable that holds state information. These formats are as follows:

Widget types	Widget variable values
Text box	The actual text (including newlines)
List box	List of selections
Combo box	
Radio button	
Check box	

Loading Complex Widgets

Several widgets need two pieces of information: a list of choices, and one or more selections from the list. For example you might have a check box group with the choices `Weekdays`, `Saturday`, and `Sunday`, and want the choice `Weekdays` to be selected by default.

Choices and selections can be defined on the Edit Widget dialog box in the fields **Choices** and **Selected**. List choices (and selections) in a single string with a space used as a separator.

When using MQL, set the widget's stored database value to the following:

```
VALUE Weekdays Saturday Sunday SELECTED Weekdays
```

As it reads from the database, the system will load choices into the widget until it finds `selected`. It then assumes that the tokens that follow are default selections.

Another approach is to use a load program. By convention, the choices go into an RPE variable whose name is the same as the load program (obtained through argument 0) and the selections go into an RPE variable whose name is the same as the widget name (obtained through argument 1). Remember that the widget variable in the RPE holds its state, and the state of a complex widget is its current selections. So the load program has identified the state of the widget by loading the variable associated with the argument 1.

Using Spaces in Strings

Lists of widget choices and selections are given in a single string with spaces used as separators. This means that any choice containing spaces should be quoted.

However, due to the extensive use of the Tcl language for manipulating lists, the Tcl notation for lists is recognized by the system. Therefore curly braces can be used to quote single list items. This is true for all arguments being passed into program objects and all values returned by program objects. The system also does a better job of supporting nested use of curly braces than it does with quote characters.

Using ${} Macros

For compatibility, ${} macros continue to be supported. ${} macros are placed into the RPE so that they can be used by nested programs. Just remember to drop the "${}" characters when referencing the macro variable in the RPE.

The following macros are used in wizards:

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 command button. back The user clicked the Back command button. next The user clicked the Next command button. finish The user clicked the Finish command button. repeat The current frame has been redisplayed. status The current frame is the status frame of the wizard.

SELECTEDOBJECTS	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, right-mouse menu or Methods dialog. 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.)
	The SELECTEDOBJECTS macro can be read into a Tcl string or list variable as follows:
	`# 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]]`
	When no objects are selected, the macro is created, but holds an empty string.

Although the operating system determines the valid syntax of the commands, the typical syntax used is:

```
    command ${MACRO 1} ${MACRO 2} ...
```

Using Eval Statements

You can wrap the code in an eval statement to prevent the output window from popping up. Placing tcl code in an eval statement causes output to be suppressed when executed within Live Collaboration.

Running/Testing the Business Wizard

Any user with appropriate access can run a wizard program from within Navigator or from the system command line, including from a Windows desktop icon. A wizard can also be launched from within a program.

MQL trace can be used to test timing and debug wizard programs. For information, see the *Administration Guide : Configuring Server Diagnostics*.

A wizard can be run by any of the following methods:

- including its name in an Live Collaboration Toolset.
- executing the wizard as a Method, which requires an object as its starting point

Command Line Interface

From the system command line, the syntax of the command to run a wizard program is:

```
matrix -wizard NAME
```

where `NAME` is the name of the wizard program to launch.

For method wizards, the command line is:

```
matrix -wizard businessobject TYPR NAME REV WIZARD_NAME
```

```
where:
    TYPE NAME REV is the fully-qualified name of the business
    object.
    WIZARD_NAME is the name of the wizard program to launch.
```

When a wizard is run from the system command line, the user sees only the wizard user interface and is insulated from the rest of the user interface. The Live Collaboration application is started, and when the wizard code has been executed, the Live Collaboration application shuts down. For this reason, a Business Administrator testing the wizard throughout its creation process would probably test from within Matrix Navigator instead of using this method.

For example, a user may have to check in a report each week. A wizard could be created to gather information such as the name and type of the report and the user/department submitting it. Because the user does not need any of the other features of Live Collaboration to check in the report, the wizard could be run from the system command line or a desktop icon in Windows.

Matrix Navigator Interface

A wizard can also be run from within Navigator by either including its name in a Navigator Toolset or by executing the wizard as a Method, which requires an object as its starting point.

When a program or wizard is added as a method to a type, then that program or wizard needs a business object in order to execute even if Needs Business Object box was not enabled for that program/wizard.

You create a Toolset by selecting the wizard program you want to execute.

To create a new Toolset

1. In Matrix Navigator, select **Visuals** from the View menu. The Visuals Manager is displayed. Click the Tools tab if it is not displayed already.
2. Click or select **New** then **Tool** from the View menu. The New Toolset dialog box is displayed.
3. Enter a name for the Toolset in the **Name** text box.
4. Select the Programs tab. The available programs and wizards are displayed. Select the wizard to be added to the toolset and click **Add** to move it to the left side of the New Toolset dialog box.
5. Click **Create**.
6. To activate the Toolset, click the box to the left of it in the Tools tab of the Visuals Manager so that a check mark appears. The **Apply** or **OK** buttons will update all Live Collaboration browsers with the selected Visuals.

The Toolset button will appear in your toolbar only when the Toolset is activated.

See the *Matrix Navigator Guide : Creating a Toolset, Chapter 5* for complete information.

Debugging the Wizard

There are many programs involved when a wizard runs. If any of the helper functions fail (for example, a widget load program) then the wizard aborts. Therefore it makes sense to debug all the helper programs first, then work on the wizard program.

You may find it helpful to display the RPE from time to time. To do this, you could create a button on some frames that executes the following program:

```
list env;
quit;
```

This dumps the RPE to the Live Collaboration output window. It also helps to place this same code at the top of your wizard code during debugging.

MQL trace can also be used to test timing and debug wizard programs. For information, see the *Installation Guide : Server Diagnostics*.

Sample Programs

Prologue Example

One of the uses of a prologue program is to cause the frame to be skipped by returning a non-zero value.

This example shows prologue code that is associated with a special frame to handle expenditures over $10,000:

```
tcl;
eval {
  set status   1
  if { [mql get env ExpenditureTotal] > 10000 } {
     set status   0
  }
  exit $status
}
```

Looking at the code, we see that this frame will always be skipped (non-zero status) unless the value of the variable ExpenditureTotal is greater than 10,000.

Epilogue Example

The epilogue program is often used to undo what the prologue program did, and perform any other cleanup activity. Since data is passed between frames using RPE, you can use the epilogue program to clean up widget variables before moving on to the next frame.

For example, epilogue code to unset the mincost and maxcost variables and remove the variables from the RPE could look like this:

```
tcl;
eval {
  mql unset env mincost
  mql unset env maxcost
exit
}
```

Load Program Example

The load program can be used to define alternate values for the widget depending on the context, information gathered in a prior frame, etc.

For example, load code associated with a combo box, used for selecting a color, might look like:

```
tcl;
eval {
  set choices [mql get env 0]
  set selected [mql get env 1]
  mql set env $choices "pink velvet mauve lavender"
  mql set env $selected "velvet"
}
exit
```

In this example, environment variable “0” is the program name, and environment variable “1” is the widget name. The choices to be included in the combo box are set in the variable

$choices. The choice to be highlighted when the combo box is displayed is set in the variable $selected.

Validate Program Examples

The validate program tests the validity of the widget’s state. If the validate program returns a non-zero value, the system redisplays the frame and places focus on the offending widget. It is good style to have the validate program additionally display a message box that explains the error.

Example 1

For example, validate code associated with a text box used to get a user’s name might look like:

```
tcl;
eval {
  set nameField [mql get env 1]
  set value [mql get env $nameField]
  if { $value == "" } {
     mql warning "must supply your name"
     exit 1
  }
}
exit 0
```

In this example, the variable $nameField is set to the widget name, which the system supplies as environment variable “1”. The variable $value is set to the value of $nameField, and then checked. If $value contains a null value (no name supplied), then the validate program returns a value of 1. A non-zero exit code causes the frame to be redisplayed. A warning is displayed to the user.

The code is wrapped in an eval statement to prevent the output window from popping up. Placing tcl code in an eval statement causes output to be suppressed when executed within Live Collaboration.

Example 2

A more generic example might involve validating that a number is in a given range. In this case, the load program would be passed two arguments that represent the range limits. The code might look like:

```
tcl;
eval {
  set widgetName  [mql get env 1]
  set min         [mql get env 2]
  set max         [mql get env 3]
  set widgetValue [mql get env $widgetName]
  if { $widgetValue < $min || $widgetValue > $max} {
     mql warning "$widgetName value of $widgetValue is out of
range ($min, $max)"
     exit 1
  }
}
exit 0
```

In this example, the variable `$widgetName` is set to the widget name, which the system supplies as environment variable “1”. Environment variable “2” is the first argument from the **Input** field, and is set to the variable `min` to represent the minimum number of the range. Environment variable “3” is the second argument from the **Input** field, and is set to the variable `max` to represent the maximum number of the range. The variable `$widgetValue` is set to the value of `$widgetName`, so it can be checked. If it is less than the value of `$min` or more than the value of `$max`, the program exits with an exit code of 1. A non-zero exit code causes the frame to be redisplayed. A warning tells the user that the value is out of range.

11

Working With Programs

Overview of Programs

A *program* is an object created by a Business Administrator to execute specific commands. Programs are used:

- In format definitions for the edit, view, and print commands.
- As Action, Check, or Override Event Triggers, or as actions or checks in the lifecycle of a policy.
- To run as *methods* associated with certain object types. (Refer to *Selecting Methods* in Chapter 4 for the procedure to associate a program with a type.)
- In Business Wizards, both as components of the wizard and to provide the functionality of the wizard.

A program might execute operating system commands. This type of program is *external*. Examples are programs such as a word processor or a CAD design program which can be specified as the program to be used for the edit, view, and print commands in a format definition.

Other programs might use only MQL/Tcl commands. For example, a check on a state might verify the existence of an object using an MQL program.

Depending on how commands are executed (either from within a program object or typed directly into the MQL console) and what mode is active (MQL or Tcl) slightly different syntax may be required.

Some programs may require a business object as the *context* or starting point of the commands. An example of this is a program that connects a business object to another object.

Defining a Program

In order to run a program, a name and the program commands are required. The type that you select determines whether the code for this program is evaluated as MQL/Tcl commands or system/application commands.

To define a program

1. Click 🖥 or select **Program** from the Object>New menu.

   The New Program dialog box is displayed, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

3. Click **Create** to add the new program object to the database.

   Or

   Click **Cancel** to exit the dialog without making changes to the database.

Defining the Name

You must specify a unique name for each program that you create. You should assign a name that has meaning to both you and the user. The program name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

For example, the name might reflect an application program from which commands are used. The program name will appear whenever the program is listed in a window.

Note: If you rename a program, it may become unavailable within certain Live Collaboration features. For example, if you rename a program that is part of a toolset, the program will need to be added to the toolset again.

Assigning an Icon

You can assign a special icon to the new program. Icons help users locate and recognize items. When assigning an icon, you must select a GIF file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

Descriptions help to clarify the purpose of the program. The description provides general information to both you and other Business Administrators about the commands associated with this program and the overall function of the program. For example, assume you want to write descriptions for two word processing programs. For the program named “BestWord” you might assign a description such as:

```
Use to launch the BestWord word processing application
```

There is no limit to the number of characters you can include in the description. However, keep in mind that the description is displayed when the mouse pointer stops over the program in the Program chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Assigning a User

You can assign a user to a program object such that when other users execute the program, they get the accesses available to the user specified in the program definition. This access inheritance includes both business and system administrative access, as well as any business object access the context user didn't already have that the program user does.

Only one user is ever associated with a program. For nested programs, the user’s access from the first program is inherited only if the called program has no associated user of its own. If the called program does have a user, then that user’s accesses are made available instead. Once the called program returns to the calling program, the latter's user is restored as the person whose accesses are added to the current context. Thus, only one person's accesses are ever added to the current context (not including access granted on a business object).

JPOs and Program User Access

JPO execution has special rules. Consider this sample code for JPO B:

```
public class ${CLASSNAME} extends ${CLASS:A} implements
${CLASS:C}
{
    public int mxMain(Context ctx,String[] args)
   {
      ${CLASS:D} dObject = new ${CLASS:D}(ctx);
      dObject.methodOfD();

      methodOfA(ctx);

      retVJPO.invoke(context, "D", null, "mxMain", null);

_mql = new MQLCommand();
_mql.executeCommand(ctx, "execute program D");
   }
}
```

The above program can be run in any of the following manners:

1. JPO B extends JPO A and a method of A is run on an object of type B as shown by methodOfA.
2. JPO B implements JPO C. References to C objects really execute a different program object.
3. JPO B runs a method of JPO D without using JPO.INVOKE as shown with dObject.
4. JPO B uses JPO.INVOKE to run JPO D.
5. JPO B uses MQLCommand to run “execute program <program name>” as shown with _mql.

In the first 3 cases, execution is handled solely by the JVM, so that Live Collaboration is never aware of when the methods of another JPO get invoked and returned. In these cases, the user of program B will remain in effect and the users, if any, of programs A, C and D, are ignored. In cases 4 and 5, execution goes through the kernel code and the programs are invoked as program objects, not just Java code by the JVM, and so the usual rules apply.

A note about program access

In native apps (MQL or Matrix), you cannot see the code in a hidden program unless you have "admin program" access. However, in a server environment this restriction does not apply .The reasoning is that you must be able to restrict end-users of rich clients from seeing "secret" programs, including those in which passwords are hidden. But in a server environment:

- app-level code needs to be able to use such secret information and;
- ordinary end-user are not supposed to have such direct access to MQL commands and so it should remain secure.

Defining a Type

You can specify the type of program as **Java**, **MQL** or **External**.

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.

An MQL program can be run from any machine with Live Collaboration installed; it is not necessary for MQL to be available on the machine. The default type is MQL.

An external program consists of commands that are evaluated by the command line syntax of the machines from which they will be run. When creating external programs, remember that the commands that you enter will be evaluated at each user’s workstation as if they were being typed at the operating system’s command prompt. Be sure that the users have the appropriate application files available from their workstation. External program objects can also be defined as “piped,” providing a built-in MQL command line service to handle standard input and output. Refer to *Defining the Piped Option* for more information.

Since MQL can be launched from a command line, MQL code could be specified in an external program. This would spawn a separate MQL session that would run in the background. In this case, MQL would have to be installed on every machine that will run the program. For more information, refer to the *MQL Guide* and the *Configuration Guide* for more information.

Defining the Execute Option

Specify when the program should be executed. Select **Immediate** or **Deferred**.

Immediate execution means that the program runs within the current transaction, and therefore can influence the success or failure of the transaction, and that all the program’s database updates are subject to the outcome of the transaction.

Deferred execution means that the program is cued up to begin execution only after the outer-most transaction is successfully committed. A deferred program will not execute at all if the outer transaction is aborted. A deferred program failure affects only the new isolated transaction in which it is run (the original transaction from which the program was launched will have already been successfully committed).

However, there are a number of cases where deferring execution of a program does not make sense (like when it is used as a trigger check, for example). In these cases the system will execute the program immediately, rather than deferring it until the transaction is committed.

There are four cases where a program’s execution can be deferred:

- Stand-alone program

- Method
- Trigger action
- State action

There are six cases where deferred execution will be ignored:

- Trigger check
- Trigger override
- State check
- Range program
- Wizard frame prologue/epilogue
- Wizard widget load/validate

There is one case where a program's execution is always deferred:

- Format edit/view/print

A program downloaded to the Web client for local execution (see *Defining the Downloadable Option*) can be run only in a deferred mode. Therefore, selecting the **Downloadable** option automatically sets the Execute option to Deferred.

Specifying the Need for a Business Object

You can specify that the program must function with a business object by checking the **Needs Business Object** option. For example, you would select this option if the program promotes a business object. If, however, the program creates a business object, the program is independent of an existing object, and this option would not apply.

When a user selects an object and then requests its methods, the programs displayed are associated with the type of the selected object. Live Collaboration runs a program that requires a business object with the selected object as the starting point. If, however, a method does not use a business object, the selected object would not be affected.

If not set, some macros, such as OBJECTID, will not be available.

The `doesneedcontext` selectable is available on programs to determine this setting in an existing program.

Defining the Downloadable Option

If the program includes code for operations that are not supported on the Web product (for example, Tk dialogs or reads/writes to a local file), you can check the **Downloadable** box. If this is checked, this program is downloaded to the Web client for execution (as opposed to running on the server). For programs not run on the Web product, this flag has no meaning.

Java Program Objects cannot be downloadable.

Defining the Hidden Option

You can mark the new program as “hidden” so it does not appear in the Program chooser which simplifies the end-user interface. Many programs are not intended to be executed as stand-alone programs (such as the programs that comprise a wizard) and users should not be able to view these program names in the Program chooser. Hidden objects are accessible through MQL, but printing a hidden program is not possible unless you are an Business or System Administrator.

Defining the Piped Option

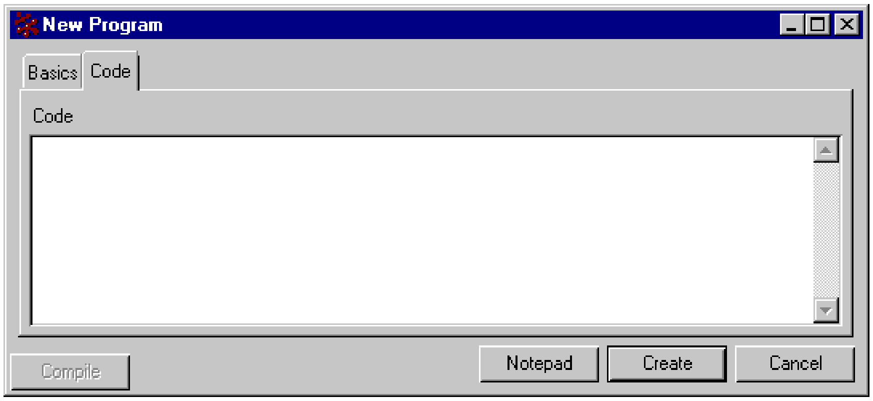
When piped is checked, External is automatically selected. The piped service is not available to JPO or MQL program objects. Execution can be immediate or deferred. A piped program cannot be downloadable.

Defining Pooled Programs

Each time a program object runs Tcl code and then exits out of Tcl mode, a Tcl interpreter is initialized, allocated, and then closed. During a session, you may execute several programs, and one program may call other programs, all of which require a Tcl interpreter and therefore the overhead of its use. In an effort to optimize multiple Tcl program execution, Tcl program objects can be specified as "pooled." When such a program is first executed in a session, a pool of Tcl interpreters is initialized, one of which is allocated for the executing code. When the code is completed, the interpreter is freed up. Subsequent Tcl code that is executed in a pooled program during the session will use an interpreter from the already initialized pool.

Defining the Code

To define the program code, click the **Code** tab. The following dialog is displayed:



Program code can be entered into the **Code** tab of the program object using the user interface, or it can be written in an external editor and automatically imported into the program object.

The "Notepad" button is included on the Code tab only when the MX_EDITOR and MX_EDITOR_PATH variables are set in the initialization (.ini) file. Refer to the *Installation Guide* for more information.

As long as you access the external editor from this dialog, Live Collaboration maintains the synchronization between the saved file and this dialog. If the code is written using the "Notepad" button (or whatever you specify for the button text with MX_EDITOR), when the external editor is closed, the contents of the saved file is shown in the Code text box.

Legal characters in XML are the tab, carriage return, line feed, and the legal graphic characters of Unicode, that is, #x9, #xA, #xD, and #x20 and above (HEX). Therefore, other characters, such as those created with the ESC key, should not be used for ANY field in Live Collaboration, including business and administrative object names, description fields, program object code, or page object content.

Creating the Program

After you enter all appropriate information on the New Program window, click **Create**. A system message tells you the program is being created.

If you see an error message, correct the definition as required and click **Create** again.

Modifying a Program Definition

After you establish a program definition, you can add or remove defining values. For information about modifying program definitions, refer to *Modifying an Administrative Object* in Chapter 1.

When modifying a program that is used to launch an application, consider upward and downward compatibility between software versions.

12

Working With Commands and Menus

Introduction to Configurable Application UI Components

Administration objects are available to control the user interface (UI) of 3DEXPERIENCE products. They provide an application programming environment that allows solutions to be easily and consistently tailored to user requirements. The following administration object types have been instantiated in Business Process Services and used in 3DEXPERIENCE products:

- command
- menu
- inquiry
- table
- Web form
- channel
- portal

The UI for 3DEXPERIENCE products is composed of many pieces. The MyDesk menu, Action menu, navigation trees, tables, forms, toolbars, and action bars have been created using the classic Live Collaboration dynamic modeling approach; command and menu objects are first created using Business Modeler or MQL and then referenced in the application's JSPs. Commands are child objects of menus—commands are created first and then added to menu definitions, similar to the association between types and attributes. Commands are also referenced in Channel objects, which are in turn used in Live Collaboration portal definitions. Inquiries are used to gather a list of business objects that can then be displayed in a configurable table. Changes made in any definition are instantly available to the applications that use it.

The use of the term "Portal" in this chapter refers to the administration object, and not to the broader internet definition described in JSR 168. In headings in the documentation we use the terms Live Collaboration portal and Public portal when a discernment needs to be made.

Commands, as well as table columns, channels, portals, and Web form fields can be user-based; that is, only shown to particular persons, roles, groups, or associations. For example, a menu command might only be available to a particular person who happens to be the administrator. Or a user defined as a Buyer might be shown a field in a Web form that is not seen by a Supplier user. When no users are specified in the command, table column, or Web form field, they are globally available to all users.

This chapter provides information on how to create menu and command objects in Business Modeler. Refer to the following chapters for information on the other configurable UI administrative objects:

Working With Inquiries and Tables

Working With Web Forms

Working with Channels and Portals

Overview of Dynamic User Interface

The dynamic user interface lets you:

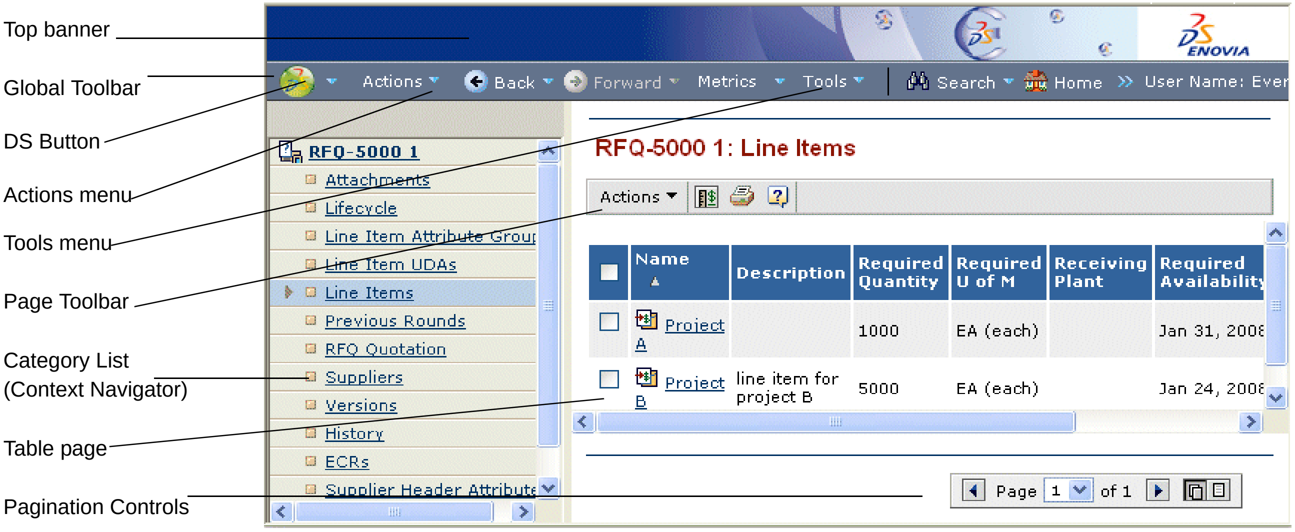
- Configure existing applications that are built using the dynamic user interface.
- Add menus and features to existing applications.
- Create a custom application that contains the interface.

Using the dynamic user interface, you accomplish all this by modifying administration objects instead of changing code in JavaServer Pages or JavaBeans.

This section contains overview information for the dynamic user interface. For details about how to build the components, see the *Configuration Guide*.

Components of the Dynamic User Interface

This graphic shows the main elements within the Business Process Services dynamic user interface. Each component is described in subsequent sections.



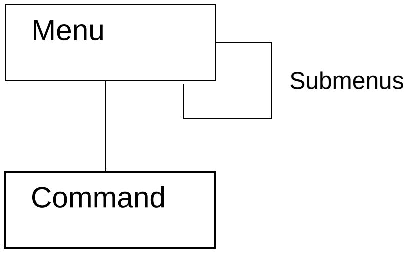
RFQ-5000 1: Line Items

Name	Description	Required Quantity	Required U of M	Receiving Plant	Required Availability
Project A		1000	EA (each)		Jan 31, 2008
Project B	line item for project B	5000	EA (each)		Jan 24, 2008

Tools for Building the User Interface

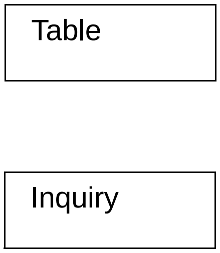
The tools that let you build and configure the dynamic user interface consist primarily of menu, command, table, and inquiry administrative objects, which are defined using Business Modeler or MQL.

Menus include the DS Button, Actions and Tools menus, the Page Toolbar, Context Navigator trees for object types, and action bars. Commands include links on menus, tools on the toolbar, action links on pages, categories in navigation trees, and action links in action bars. As shown below, a menu can have submenus, and menus and submenus can have commands.



You define commands separately from menus so a single command can be used in multiple locations. For example, a command could be used in the My Desk menu for two different applications and in the navigation tree for several different object types.

Tables are the columns displayed on table pages and inquiries define the objects included in a table.



Another set of tools for building the dynamic interface include JavaServer Pages (JSPs). One JSP page, emxTree.jsp, controls the tree display. The Business Process Services also includes JSPs for standard pages that are used in every 3DEXPERIENCE product, such as a logout, change password, and help about page.

The Business Process Services installs the tools needed to build the common UI elements needed for all applications, such as the toolbar, My Desk and Actions menus, and common commands, such as logout. Each application installs the specific menus and commands needed for it.

The sections that follow describe how to use these tools to build custom applications using the same user interface model used by 3DEXPERIENCE products. For details on creating and configuring the administrative objects, see the appropriate chapters in this document.

Configurable Pages

These are the configurable pages.

- emxTree.jsp
- emxTable.jsp

Standard Commands

The framework Business Process Services includes some standard commands that all 3DEXPERIENCE products use. Because they are used by all applications, they are assigned to the Toolbar menu.

- Change Password
- Home page
- Home page preference
- Application Menu

- Logout
- Help About
- IconMail
- Help

Naming Conventions for UI Administrative Objects

The following table lists naming conventions used for UI objects installed with the Business Process Services. 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 table on the following page. If you create custom objects, create your own abbreviation based on your company name. This applies to anyone creating custom objects, including customers, partners, and 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) and everyone should use these objects. Do not include spaces in the names for UI administrative objects.

Admin Object Type, Usage	Convention	Examples
Menu objects for menu types (installed with Business Process Services)	Menu type name	My Desk Actions Toolbar
Menu objects for application submenus within a menu type (installed with applications; no submenus for toolbar)	3-letter standard abbreviation for application name (see table below), followed by the menu type with no spaces.	ENCMyDesk TMCActions
Menu objects for root node of trees	Symbolic name of the type the tree is for or the type's parent type.	type_Part type_DrawingPrint Default Tree
Menu objects for action bars	3-letter standard abbreviation for application name (see table below), followed by the action bar name, and then the menu type, which is Action Bar in this case.	ENCBOMListActionBar TMCWorkspacesActionBar
Command objects for toolbar tools, menu options, tree categories, and action links on pages	3-letter standard abbreviation for application name (see table below), followed by the action or feature name, followed by the type of command (Actions, MyDesk, Tree, Toolbar, TreeCategory, or ActionBar).	ENCBOMMyDesk ENCCreatePartActions TMCEditProfileToolbar AEFLogoutToolbar AEFHistoryTreeCategory ENCDeleteSelectedActionLink
Table objects	3-letter standard abbreviation for application name (see table below), followed by the feature name.	ENCBOMList TMCWorkspaces SCSBuyerDesks
Inquiry objects	3-letter standard abbreviation for application name (see table below), followed by the list name or the feature the inquiry is for.	ENCCBOM TMCPersons SCSPackages

Application	3-Letter Abbreviation Used for Objects (do not use these abbreviations when creating custom objects)
Application Exchange Framework and Common Components (part of Business Process Services)	AEF APP
Engineering Central	ENC
Program Central	PMC
Sourcing Central	SCS
Specification Central	SPC
Supplier Central	SUP
Variant Configuration Central	FTR
Team Central (part of Business Process Services)	TMC
Library Classification	LIB
Materials Compliance	MCC

Using Macros and Expressions in Configurable Components

Many strings used in the definition of configurable components (such as label values, hrefs, and settings) can contain embedded macros and select clauses. The ${} delimiters identify macro names. Macros are evaluated at run-time. New macros for configurable components are available for directory specification. Some existing macros are also supported. See *Directory Macros* and *Select Expression Macros*.

Strings defined using macros and expressions cannot be internationalized.

Some strings can also include select clauses which are evaluated against the appropriate business object at run-time. The $<> delimiters identify select clauses. Because the select clauses will generally use symbolic names, the clauses will be preprocessed to perform any substitutions before submitting for evaluation. The following example shows a macro being used in the href definition and another macro being used in the Image setting, as well as a select clause being used in the label definition of a tree menu (associated with a LineItem object):

```
MQL<2>print menu type_LineItem;
menu type_LineItem
description
label '$<attribute[attribute_EnteredName].value>'
href '${SUITE_DIR}/emxQuoteLineItemDetailsFS.jsp'
setting Image value ${COMMON_DIR}/iconSmallRFQLineItem.gif
setting Registered Suite value SupplierSourcing
children
command SCSAttachment
command SCSAttributeGroup
command SCSHistory
command SCSSupplierExclusion
command SCSUDA
nothidden
property original name value type_LineItem
property installed date value 02-28-2002
property installer value MatrixOneEngineering
property version value 11-0-0-0
property application value Sourcing
created Thu Feb 28, 2002 11:12:34 AM EST
modified Thu Feb 28, 2002 11:12:34 AM EST
```

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

```
MQL<3>print menu type_Company;
menu type_Company
description
label ’${NAME}’
href ’${SUITE_DIR}/emxTeamCompanyDetailsFS.jsp’
setting Image value ${COMMON_DIR}/
iconSmallOrganization.gif
setting Registered Suite value Team
children
command TMCBusinessUnit
command TMCLocation
command TMCPeople
nothidden
property original name value type_Company
property installed date value 02-28-2002
property installer value MatrixOneEngineering
property version value 11-0-0-0
property application value Team
created Thu Feb 28, 2002 11:31:57 AM EST
modified Thu Feb 28, 2002 11:31:57 AM EST
```

When using macros, surround it with quotes to ensure proper substitution for values that contain spaces.

Directory Macros

The following table provides the list of directory-specific macros that can be used in parameters and settings for configurable components.

Macro Name	Use to:
${COMMON_DIR}	Substitute the “common” directory below DOCROOT/ematrix directory. The substitution is done with reference to any application-specific directory and it is relative to the current directory.
${ROOT_DIR}	Substitute the “ematrix” directory. The substitution is done with reference to any application-specific directory below DOCROOT/ematrix and it is relative to the current directory.
${SUITE_DIR}	Substitute the application-specific directory below the DOCROOT/ematrix directory. The substitution is done based on the suite (application) to which the command belongs and it is relative to the current directory.

Select Expression Macros

Select expression macros are defined as $<SELECT EXPRESSION>, where the select expression can be any valid MQL select statement. Select expression macros can be used in labels for configurable components and in expression parameters. These 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[attribute_Originator].value>
- $<attribute[FindNumber].value>
- $<from[relationship_EBOM].to.name>

Defining a Command Object

As a Business Administrator, you can create new command objects if you have the Menu administrative access. Commands can be used in any kind of menu in a JSP application. Commands may or may not contain code, but they always indicate how to generate another Web page.

To define a command object

1. Click or select **Command** from the Object>New menu.

   The New Command window displays, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Name

You must specify a unique name for each command that you create. You should assign a name that has meaning to both you and the user. The name you choose is the name that will be referenced to include this command within a menu.

You cannot have both a command and a menu with the same name.

Assigning an Icon

You can assign a special icon to the new command. Icons help business administrators locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

The icon assigned to a command is used only within Business Modeler and not in the Web page that contains the command.

Defining a Description

Type a description for the command object. The description provides general information for you and the user about the function, use, or content of the command.

The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description.

For example, if you were defining a command named "Cost Evaluator" you might write a Description clause similar to one of the following. Each command might differ considerably.

`Figures cost based on wholesale prices.`
`Figures cost based on discounted prices`
`Figures monthly costs for field testing of Widget B`

Defining a Label

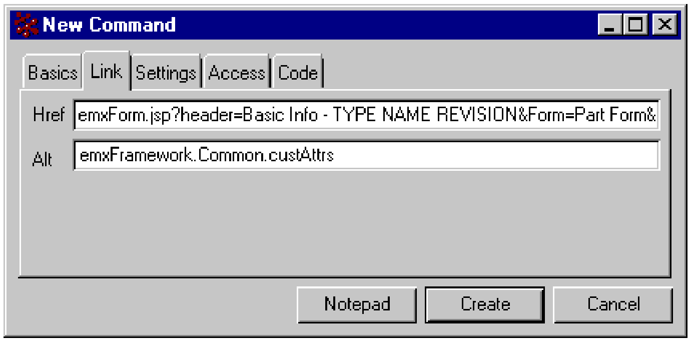
Type a label to appear in the menu in which the command is assigned. For example, many desktop applications have a File menu with options labeled "Open" and "Save."

Defining the Hidden Option

Hidden objects are not displayed to Matrix Navigator users. Since there is currently no window that shows commands in Matrix Navigator, the hidden flag is not used at this time.

Defining the Link

Select the **Link** tab to provide information to be used when the command is selected from an application page:



Href: emxForm.jsp?header=Basic Info - TYPE NAME REVISION&Form=Part Form&

Alt: emxFramework.Common.custAttrs

The information provided on the link tab is used to supply link data and alternative text to the href and alt HTML commands on the Web page that references the command. Both fields are optional, but generally an href value is included.

Href

Type a string for the Href field to be used to provide link data to the JSP. When commands are accessed from a JSP page, the Href link is evaluated to bring up the next page. The Href string generally includes a fully-qualified JSP filename and parameters. Refer to *Using Macros and Expressions in Configurable Components* for more details.

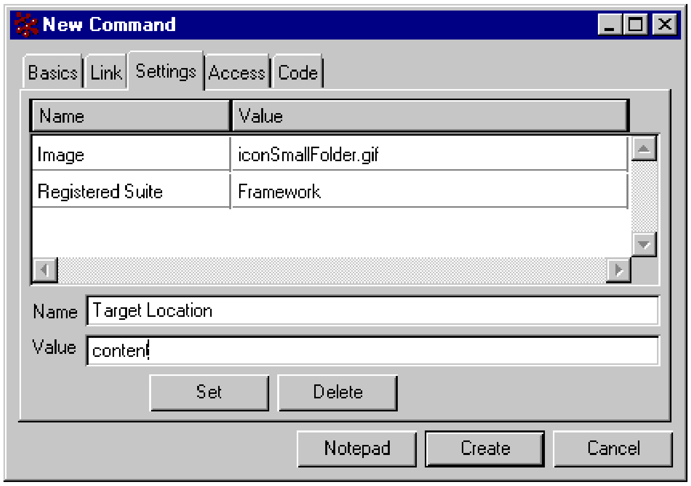
Assigning an href to a link can create problems if a user clicks the same link twice when initiating such actions as changing an object's state or submitting a form. The reason for this is as follows. An href assigned to a link is considered a server request, even if it is a JavaScript command. Whenever the browser detects a server request, the browser stops processing the current request and sends the new request. Therefore, when a user first clicks on an href link, the request is processed, and, typically, a JSP page starts executing. If, during this time, a user clicks the same link again, the first request is interrupted before completion and the new request is processed instead. To avoid this scenario, you can set the href to "#" and use the onclick event instead. The generic code for this is: `<a href="#" onclick="submitForm()">`.

Alt

Type any alternate text into the **Alt** field. This text is displayed until any image associated with the command is displayed and also as "mouse over text."

Defining the Settings

Select the **Settings** tab to provide any name/value pairs that the command may need.



Name	Value
Image	iconSmallFolder.gif
Registered Suite	Framework

Name: Target Location

Value: content

To add a setting, enter a **Name** and **Value** and click **Set**. You can remove a setting by selecting its name and clicking **Delete**.

Settings are general name/value pairs that can be added to a menu as necessary. They can be used by JSP code, but not by Hrefs on the Link tab, or on the Code tab. For example, an image setting with the image name can be specified to display when the command is used in a toolbar. Refer to the *Configuration Guide* for appropriate settings for the type of command you are creating. Also refer to *Using Macros and Expressions in Configurable Components* for more details.

Configuring the History Bit Setting

The history bit setting can be configured on commands for events in *Live Collaboration*. This setting must be configured on commands defined for events to which users can subscribe and receive notice about through the emxTransactionNotificationUtil JPO. This JPO can be called through the Trigger Manager and is invoked only after a transaction completes, regardless of how many events have occurred inside that transaction. See the *Configuration Guide: About Triggers.*

To configure the history bit setting

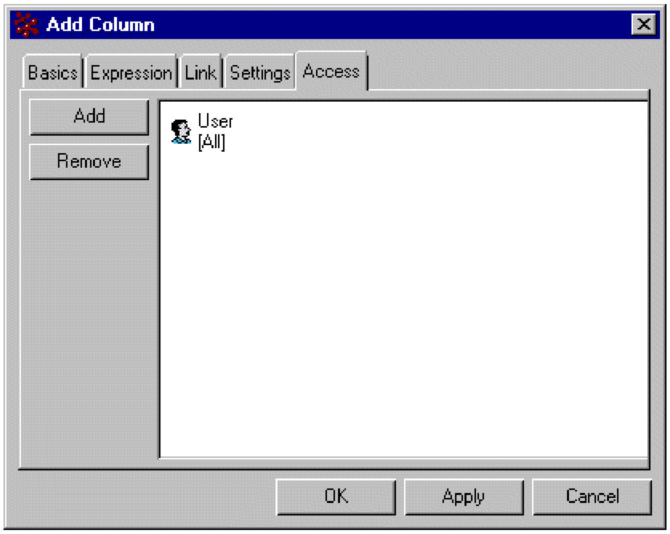
1. Find the command or event that can be subscribed to by users.
2. Select the **Settings** tab to provide any name/value pairs that the command may need.
3. Add the setting called "History Bit."
4. Add the value for the History Bit setting. The value for this setting should match the history action returned from the event. For example: for the event called PartRevisedEvent, the $TRANSHISTORY macro returns a history action called revisioned. The value for history bit in this case is "revisioned." The following are possible history bit values:

approve	checkout	disconnect
change name	connect	grant
change owner	copy	modify

change policy	create	lock
change type	delete	promote
checkin	demote	revisioned
	revoke	unlock

Defining Access

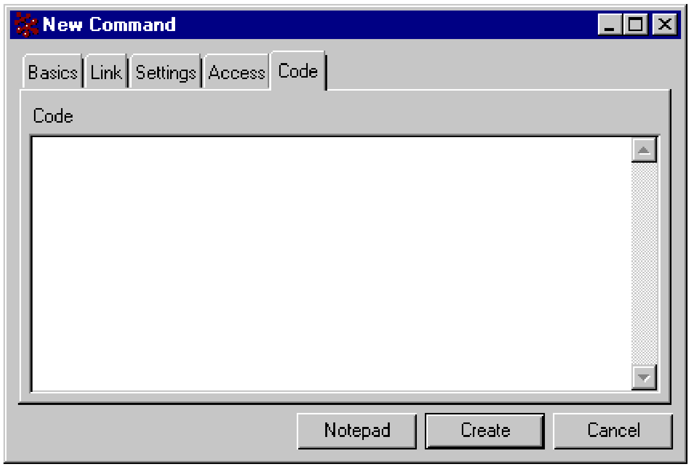
Select the Access tab to define the users allowed to see the command.



Any number of roles, groups, persons, and/or associations can be added to the command (role-based in this case includes all types of users and is not limited to only Live Collaboration roles). User [All] is specified by default. Click **Add** and a list of all existing (and not hidden) users is displayed. Limit access by selecting those specific users who will be allowed access to the command and click **OK**. Select a user and click **Remove** to remove a user's access to the command.

Assigning Code

Select the Code tab to add JavaScript code to the command.



When commands are accessed from a JSP page, the href link is evaluated to bring up the next page. Commands only require code if the href link references JavaScript that is not provided on the JSP. The JSP must provide logic to extract the code from this field in order for it to be used. None of the commands provided by the applications or Business Process Services use the code field.

You can type in the Code entry area directly or it can be written in an external editor and automatically imported into the command object. The "Notepad" button is included on the Code tab only when the MX_EDITOR and MX_EDITOR_PATH variables are set in the initialization (enovia.ini) file. Refer to the *Installation Guide* for more information.

As long as you access the external editor from this dialog, Live Collaboration maintains the synchronization between the saved file and this dialog. If the code is written using the "Notepad" button (or whatever you specify for the button text with MX_EDITOR), when the external editor is closed, the contents of the saved file is shown in the Code text box.

Creating the Command

After you enter all appropriate information on the New Command window, click **Create**. If you see an error message, correct the definition as required and click Create again.

Assigning access to a command

The user clause takes a list of roles, groups, persons, and/or associations, or it can take the keyword `all` to make the Command available to all users. Giving no user clause is the same as `all`.

Associating code with a command

The code clause is used to enter JavaScript code as a string. Alternatively, you can use the file clause to reference a file on disk from which to fetch the JavaScript code.

The code clause should only be included if the href clause references JavaScript that is not included on the page.

Defining a Menu Object

As a Business Administrator, you can create new menu objects if you have the Menu administrative access. Menus can be used in custom Java applications. Before creating a menu, you must define the commands that it will contain. Menus can be designed to be toolbars, action bars, or drop-down lists of commands.

To define a menu object

1. Click ▤ or select **Menu** from the Object>New menu.

   The New Menu window displays, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Name

You must specify a unique name for each menu that you create. You should assign a name that has meaning to both you and the user. The name you choose is the name that will be referenced to include this menu within another menu or application.

You cannot have both a menu and a command with the same name.

Assigning an Icon

You can assign a special icon to the new menu. Icons help business administrators locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

The icon assigned to a menu is used only within Business Modeler and not in the Web page that contains the menu.

Defining a Description

Type a description for the menu object. The description provides general information for you and the user about the function, use, or content of the menu.

The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description.

For example, if you were defining a menu named “Tools” you might write a Description clause similar to one of the following. Each menu might differ considerably.

`Drawing Tools`
`Tools to figure cost`
`Shortcut Tools`

Defining a Label

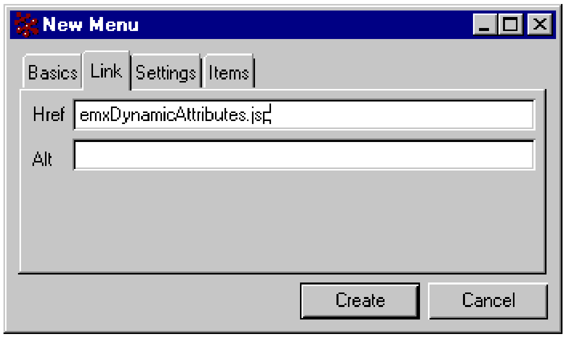
Type a label to appear in the application in which the menu is assigned. For example, many desktop applications have a File menu.

Defining the Hidden Option

Hidden objects are not displayed to Matrix Navigator users. Since there is currently no window that shows menus in Matrix Navigator, the hidden flag is not used at this time.

Defining the Link

Select the **Link** tab to provide information to be used when the menu is selected from an application page:



Href emxDynamicAttributes.jsp

Alt

The information provided on the link tab is used to supply link data and alternative text to the href and alt HTML commands on the Web page that references the menu.

Href

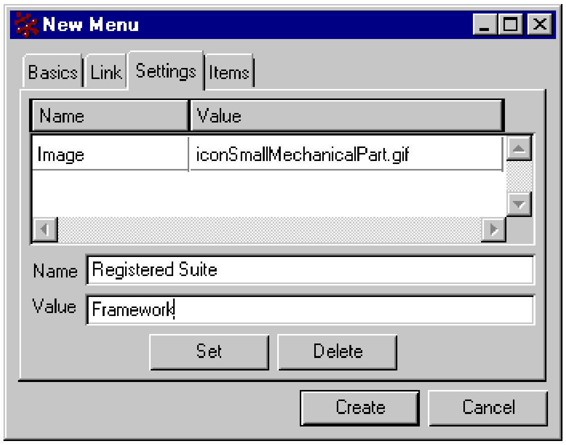
Type a string for the **Href** field to be used to provide link data to the JSP. The Href link is evaluated to bring up another page. Many menus will not have an Href value at all. However, menus designed for the “tree” menus require an Href because the root node of the tree causes a new page to be displayed when clicked. The Href string generally includes a fully-qualified JSP filename and parameters, which can contain embedded macros and expressions for mapping to database schema. Refer to *Using Macros and Expressions in Configurable Components* for more details.

Alt

Type any alternate text into the **Alt** field. This text is displayed until any image associated with the menu is displayed and also as “mouse over text.”

Defining the Settings

Select the **Settings** tab to provide any name/value pairs that the menu may need.



Name	Value
Image	iconSmallMechanicalPart.gif

Name: Registered Suite

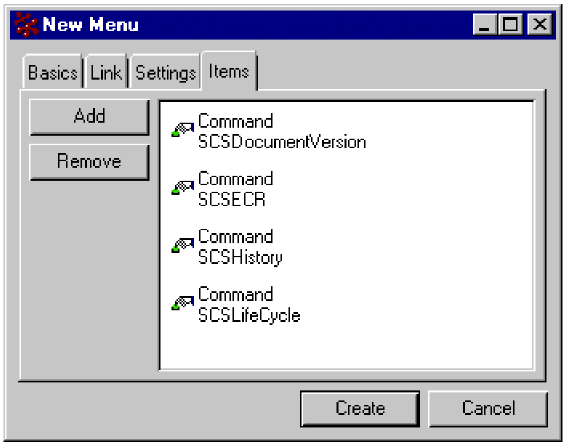
Value: Framework

To add a setting, enter a **Name** and **Value** and click **Set**. You can remove a setting by selecting its name and clicking **Delete**.

Settings are general name/value pairs that can be added to a menu as necessary. They can be used by JSP code, but not by hrefs on the Link tab. For example, an image setting with the image name can be specified to display when the menu is used in a toolbar. Refer to the *Configuration Guide* for appropriate settings for the type of menu you are creating. Also refer to *Using Macros and Expressions in Configurable Components* for more details.

Assigning Menu Items

Select the **Items** tab to add commands and other menus to the menu.



Command SCSDocumentVersion

Command SCSECR

Command SCSHistory

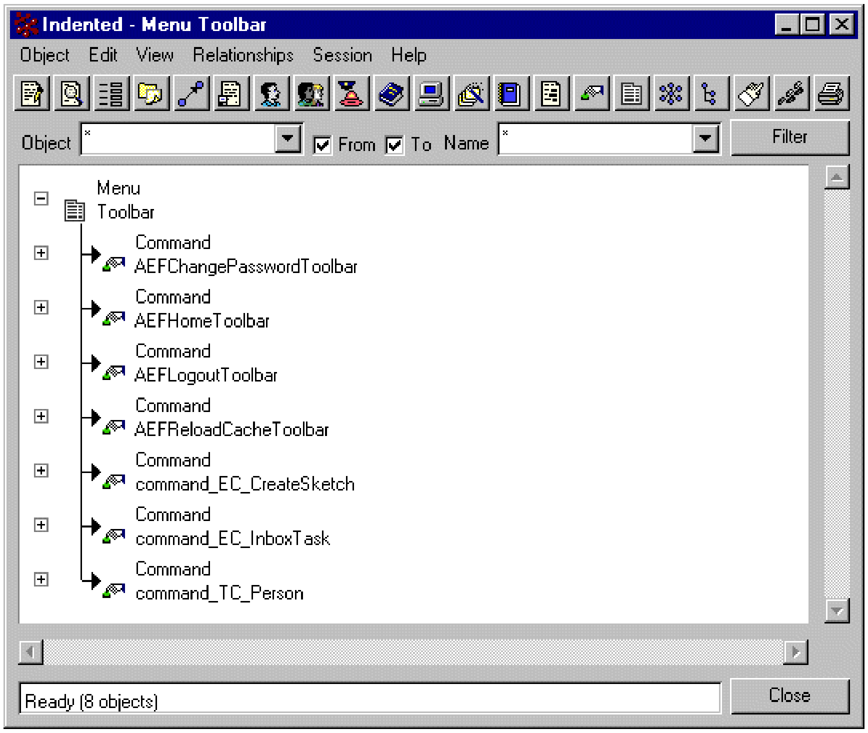
Command SCSLifeCycle

Click **Add** to display a list of all existing commands and menus. Select specific commands and/or menus to be added to the menu you are creating. The commands will be displayed in the order in which they appear in the Items tab. You can drag commands and menus to rearrange their order. Select a command or menu and click **Remove** to delete it from the menu.

Creating the Menu

After you enter all appropriate information on the New Menu window, click **Create**. If you see an error message, correct the definition as required and click Create again.

You can see the Items included by selecting the menu and clicking or .



Menu
Toolbar

Command
AEFChangePasswordToolbar

Command
AEFHomeToolbar

Command
AEFLogoutToolbar

Command
AEFReloadCacheToolbar

Command
command_EC_CreateSketch

Command
command_EC_InboxTask

Command
command_TC_Person

Ready (8 objects)

In the indented view, items are displayed alphabetically, and not in the order in which they are assigned.

13

Working With Inquiries and Tables

Inquiries and System Tables defined

As described in *Working With Commands and Menus*, administration objects can be defined that control the user interface (UI) of 3DEXPERIENCE products, providing an application programming environment that allows solutions to be easily and consistently tailored to user requirements.This chapter provides information on how to create inquiry and command objects in Business Modeler.

Business administrators can create *inquiries* and *tables*. Inquiries are designed to obtain a list of business objects, which can then be loaded into a pre-defined table in an application’s page. Tables can be defined for system-wide access, instead of being associated with a person’s (or role’s) workspace, and each column has additional parameters such as link data (href and alt) that provide flexibility and creativity for JSP programmers.

Table columns, as well as Commands and Web form fields can be user-based; that is, only shown to particular persons, roles, groups, or associations. For example, a user defined as a Buyer might be shown a column in a table that is not seen by a Supplier user. When no users are specified in the command, table column, or Web form field, they are globally available to all users.

Defining an Inquiry Object

As a Business Administrator, you can create new inquiry objects if you have the Inquiry administrative access. Inquiries can be evaluated to produce a list of objects to be loaded into a table in a JSP application. In general, the idea is to produce a list of business object ids, since they are the fastest way of identifying objects for loading into browsers. Inquiries include code, which is generally defined as an MQL temp query or expand bus command, as well as information on how to parse the returned results into a list of OIDs.

To define an Inquiry object

1. Click ▤ or select **Inquiry** from the Object>New menu.

   The New Inquiry window displays, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Name

You must specify a unique name for each inquiry that you create. The name you choose is the name that will be referenced to evaluate this inquiry within a JSP.

Assigning an Icon

You can assign a special icon to the new inquiry. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

Type a description for the inquiry object. The description provides general information for you about the function, use, or content of the inquiry.

The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description.

Defining a Pattern

The **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 RPE variable or macro. Since inquiries are designed to produce a list of business objects, generally the macro that is set is OID.

When you execute a temp query in MQL, the business objects found are returned in a list that includes the type, name and revision, as well as any selectable information specified. For example, the following code:

```
MQL< >temp query bus Part * * select id dump;
```

would return a list like:

```
Part,PT-6170-01,1,21762.30027.65182.63525
Part,PT-6180-01,1,21762.30027.50161.30295
Part,PT-6190-01,1,21762.30027.56625.19298
Part,PT-6200-01,1,21762.30027.37094.65388
```

To indicate that there are four fields that will be returned, delimited with a comma, and the last field is the OID, you would use the following pattern:

```
*,*,*,${OID}
```

For an expand bus command, even more information is output before the select fields:

```
MQL< >expand bus Person "Test Buyer" - from relationship "Assigned Buyer" select
businessobject id dump |;
1|Assigned Buyer|to|Buyer Desk|Buy 001|-|37819.19807.45300.63521
```

To parse this output, you need to indicate that the first six fields, delimited by "|", should be ignored, and the seventh field is the OID. You would use:

```
*|*|*|*|*|*|${OID}
```

Defining the Format

The **Format** defines what part of the output results should be saved in the inquiry's list. It references variables or macros specified in the Pattern, and can include delimiters. For example:

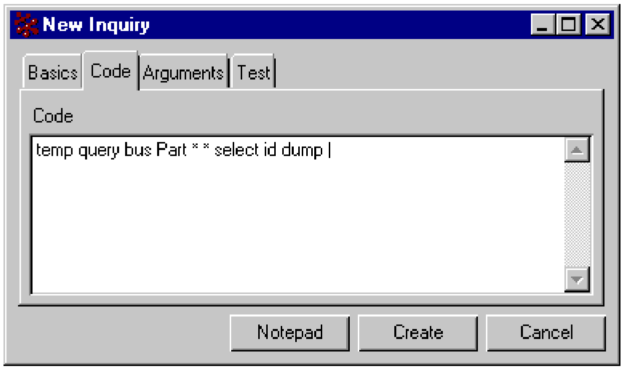
${OID}

Defining the Hidden Option

Hidden objects are not displayed to Matrix Navigator users. Since there is currently no window that shows inquiries in Matrix Navigator, the hidden flag is not used at this time.

Defining the Code

Select the **Code** tab to provide the code to be evaluated to produce a list of one or more business objects:



```
temp query bus Part * * select id dump |
```

The code provided is generally an MQL temp query or expand bus command that selects the found objects' ids. It can contain complicated where clauses as needed. For example:

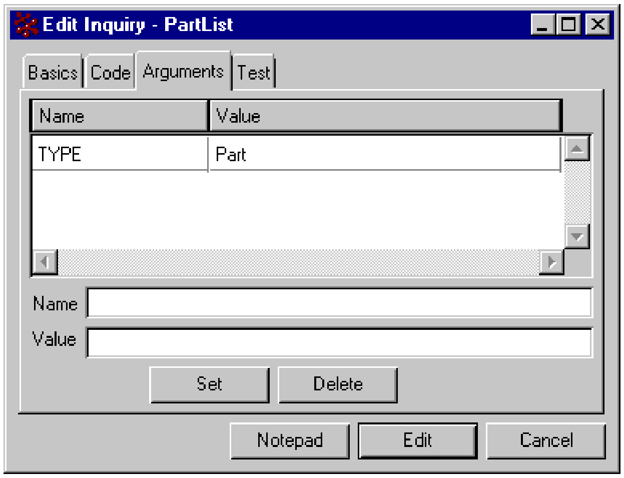
```
temp query bus "Package" * *
    where "('Project'==to[Vaulted Documents Rev2].businessobject.to[Workspace
Vaults].businessobject.type)
        &&  ('${USER}'==to[Vaulted Documents Rev2].businessobject.to[Workspace
Vaults].businessobject.from[Project Members].businessobject.to[Project
Membership].businessobject.name)"
        select id dump |;
```

When macros are included in the code (${USER} in example above), they should be surrounded by single or double quotes, in case the substitution contains a space. Quotes around both the macro in the code and the Argument when it contains a space ensures that the macro substitution is handled correctly.

You can type in the Code entry area directly or it can be written in an external editor and automatically imported into the command object. The external editor button is included on the command editor only when the editor is specified in the initialization (enovia.ini) file. Refer to *Assigning Code* in the section on commands for more information.

Defining the Arguments

Select the **Arguments** tab to provide any input arguments that the inquiry may need.



Name	Value
TYPE	Part

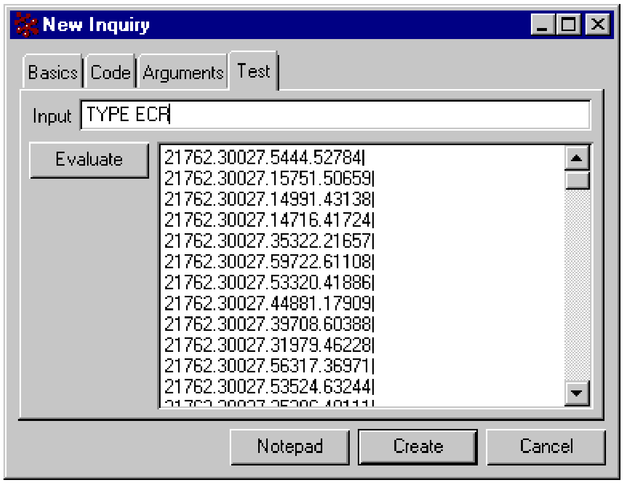
To add an argument, enter a **Name** and **Value** and click **Set**. You can remove a setting by selecting its name and clicking **Delete**. Include quotes if the value contains a space.

Quotes around both the macro in the code and the Argument when it contains a space ensures that the macro substitution is handled correctly.

Arguments are name/value pairs that can be added to an inquiry as necessary to be used by the inquiry code. Depending upon how you write the code in both the inquiry and the JSP, you may or may not use arguments.

Testing the Inquiry

You can use the **Test** tab to determine if the inquiry will parse the output as you have designed the JSP to expect to receive it.



Input: TYPE ECR

```
21762.30027.5444.52784|
21762.30027.15751.50659|
21762.30027.14991.43138|
21762.30027.14716.41724|
21762.30027.35322.21657|
21762.30027.59722.61108|
21762.30027.53320.41886|
21762.30027.44881.17909|
21762.30027.39708.60388|
21762.30027.31979.46228|
21762.30027.56317.36971|
21762.30027.53524.63244|
```

Click **Evaluate** to execute the code as specified on the Code and Arguments tabs, parse it as indicated on the Basics tab, and display the generated list.

To override any specified Arguments, or include input that the inquiry may otherwise receive from the JSP, enter name value pairs in the **Input** area. Include only a space between multiple inputs, using quotes around values that contain spaces.

Creating the Inquiry

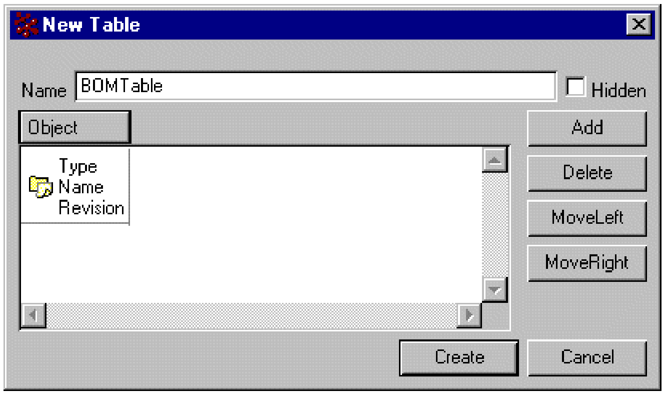
After you enter all appropriate information on the New Inquiry window, click **Create**. If you see an error message, correct the definition as required and click Create again.

Defining a Table Object

As a Business Administrator, you can create new table objects if you have the Table administrative access. System tables can be used in custom applications. Unlike the tables that users create from with Matrix Navigator, these tables are available for system-wide use, and not associated with a particular session context. Each column has several parameters where you can define the contents of the column, link data, user access, and other settings.

1. Click ▦ or select **Table** from the Object>New menu.

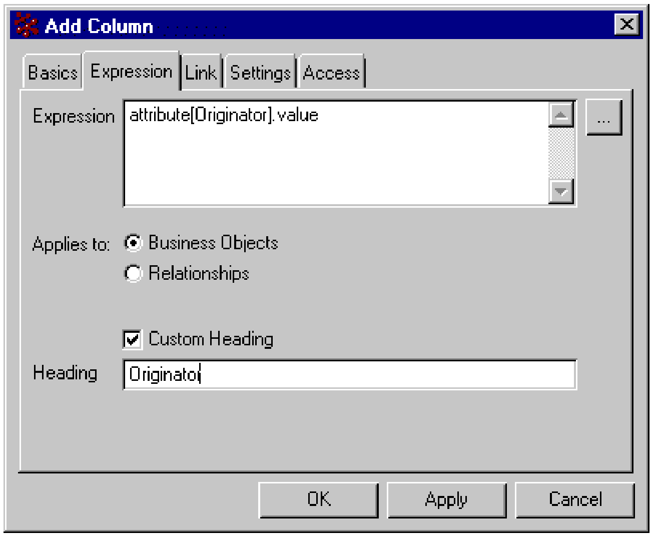
   The New Table window opens:



Name BOMTable □ Hidden

Object
Type Name Revision

2. Enter a name for the table in the **Name** text box. Table names cannot include asterisks, single quotes or commas. You must specify a unique name for each table you create. You should assign a name that has meaning to both you and the user. The name you choose is the name that will be referenced to use this table in an application.
3. Click **Add** to add a column to the table. When a column is added to a system table, it will be propagated to all tables derived from that table. The Add Column window displays, showing the **Expression** tab.



Expression attribute[Originator].value

Applies to: ● Business Objects
○ Relationships

☑ Custom Heading

Heading Originato

4. Assign values to the column’s parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

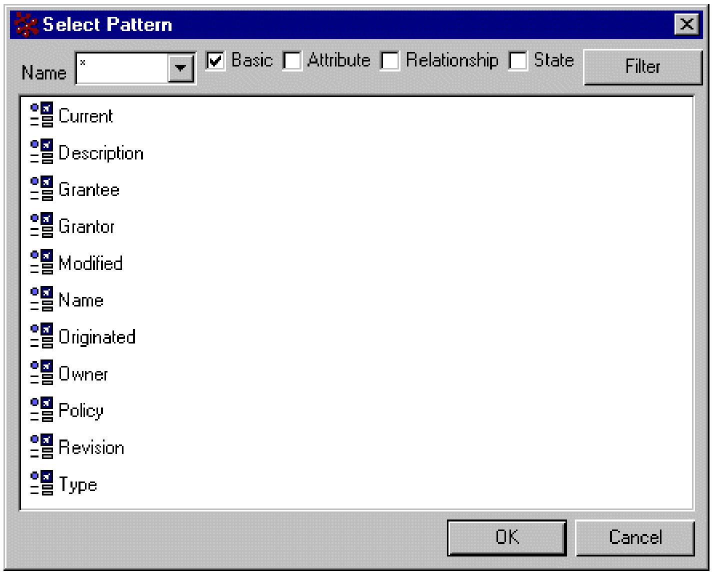
Defining the Expression

The **Expression** tab of the Add Column window allows you to define the content of the column.

To Define a Column’s Expression

1. Click ... next to the Expression field.

   The Select Pattern chooser opens and lists the basic properties of an object. You can also display attributes and relationships by selecting the check boxes and clicking **Filter**. You can filter on the Name as well.



Name * ☑ Basic ☐ Attribute ☐ Relationship ☐ State

Current
Description
Grantee
Grantor
Modified
Name
Originated
Owner
Policy
Revision
Type

2. Select the property you want to include as a table column and then click **OK**.

   The Expression tab shows the expression you just selected.

   You must define a new column for each MQL expression you want to use in the table. That is, you cannot use && (and) or || (or) qualifiers. However, you can use operators on numeric values. For example:

   ```
   attribute[Target Cost] - attribute[Actual Cost]
   ```

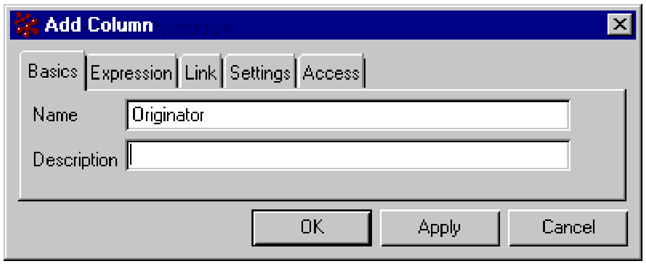
   For help formatting expressions, refer to the *Configuration Guide : About Selectables*.

3. Select whether the expression applies to **Business Objects** or **Relationships**. Expressions that apply to relationships will only be evaluated when the inquiry used to load the table includes an expand operation. For more information about applying expressions, see *Applying Expressions to Relationships vs. Business Objects*.

4. To enter a unique name for the column heading, check **Custom Heading** and enter the name in the **Heading** text box. If you do not check Custom Heading, the column heading defaults to the expression of the column.

Defining a Column’s Basics

Click the **Basics** tab to specify a name and description for the column. The column **Name** is the name that will be referenced when specifying this column in an application. You do not have to specify a column name but if you do, the name cannot be used for any other column within the table. If you do not specify a column name, the system uses the column’s database ID for its name. Keep in mind that the column name is not the text that displays in the column heading, which is specified on the Expression tab.



Name: Originator

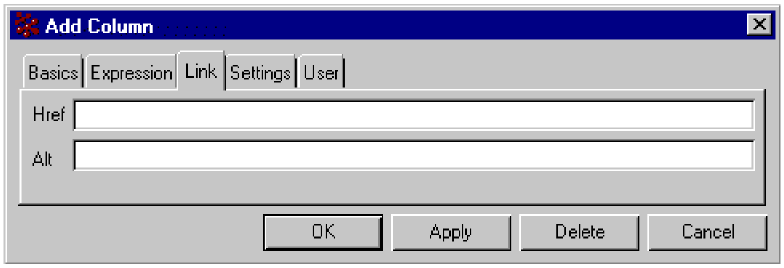
Description:

Add a **Description** for the column. The description provides general information for you about the function, use, or content of the column. The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description.

Defining the Link

Select the **Link** tab to provide information to be used when the table is selected from an application page:

:



Href:

Alt:

The information provided on the link tab is used to supply link data and alternative text to the href and alt HTML commands on the Web page that references the table.

Href

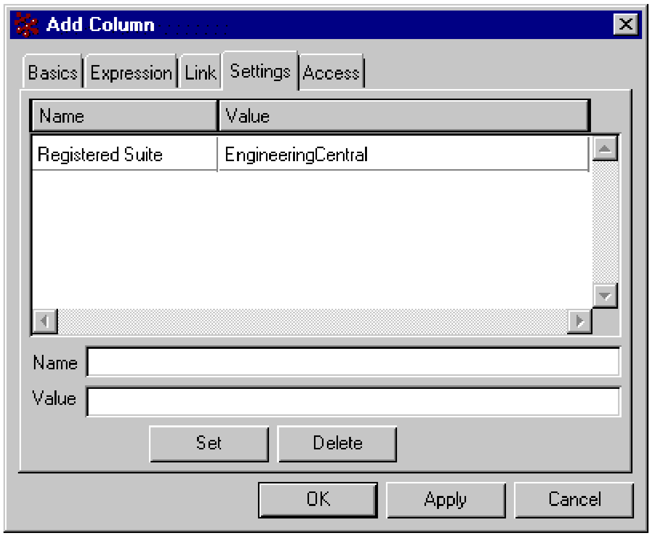
Type a string for the **Href** field to be used to provide link data to the JSP. The Href link is evaluated to bring up another page. Many tables will not have an Href value at all. The Href string generally includes a fully qualified JSP filename and parameters, which can contain embedded macros and expressions for mapping to database schema. Refer to *Using Macros and Expressions in Configurable Components* for more details.

Alt

Type any alternate text into the **Alt** field. This text is displayed until any image associated with the column is displayed and also as “mouse over text.”

Defining the Settings

Select the **Settings** tab to provide any name/value pairs that the table may need.



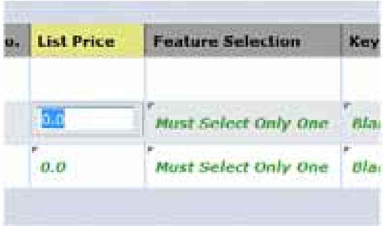
Name	Value
Registered Suite	EngineeringCentral

To add a setting, enter a **Name** and **Value** and click **Set**. You can remove a setting by selecting its name and clicking **Delete**.

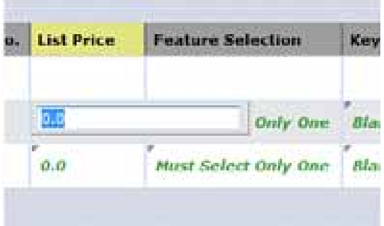
Settings are general name/value pairs that can be added to a column as necessary. They can be used by JSP code, but not by hrefs on the Link tab. For more information, see *Using Macros and Expressions in Configurable Components*.

The **Auto Fit Cell** property limits the width of input box controls that are used for entering numbers or text in a structure browser:

- Setting its value to **true** constrains the input box to the width of the cell, preventing it from overlapping the adjacent cell.



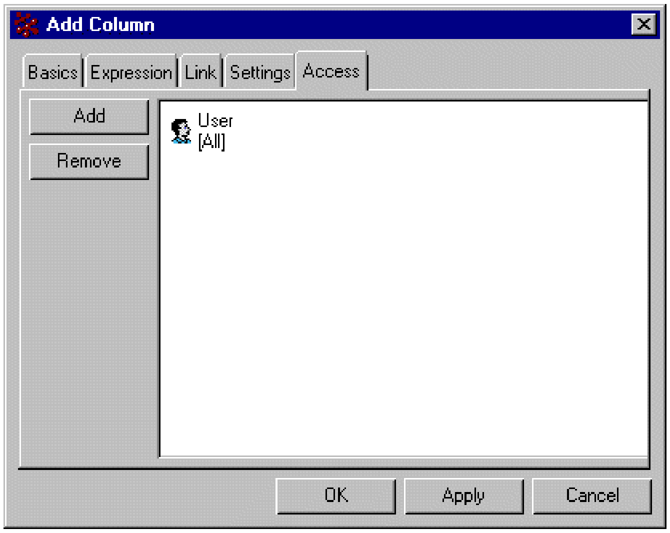
- Setting its value to **false** (the default) allows the input box to exceed the width of the cell, possibly overlapping the adjacent cell.



This property applies only to input boxes within a structure browser. It does not apply to choosers, date pickers, list boxes, check boxes, or any other type of input control.

Defining Access

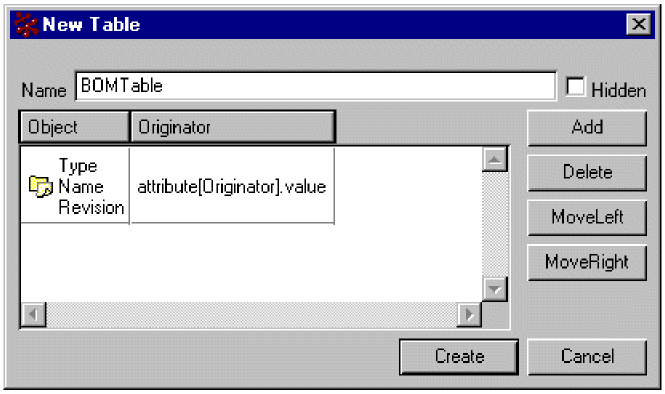
Select the Access tab to define the users allowed to see the column.



Any number of roles, groups, persons, and/or associations can be added to the column (role-based in this case includes all types of users and is not limited to only Live Collaboration roles). User [All] is specified by default. Click **Add** and a list of all existing (and not hidden) users is displayed. Limit access by selecting those specific users who will be allowed access to the column and click **OK**. Select a user and click **Remove** to remove a user’s access to the column.

Creating the Column and Table

After you enter all appropriate information on the Expression window, click **Create**. The column is added to the table. Repeat the process for all other columns.



Name: BOMTable
□ Hidden

Object	Originator
Type Name Revision	attribute[Originator].value

Hidden objects are not displayed to Matrix Navigator users. Since there is currently no window that shows business tables Matrix Navigator, the hidden flag is not used at this time.

You can rearrange and delete columns by selecting a column (not in the heading, but in the area that displays the expression) and clicking **Move Right**, **Move Left**, and **Delete**. You cannot move or delete the first column, Object. Edit a column by double-clicking on its heading. When you are satisfied with the table click **Create**.

Applying Expressions to Relationships vs. Business Objects

How you choose to apply a column expression depends on the information you want, how you define the expression, and how you want the information to display. You can get the same information into a table by defining a slightly different expression and applying it to relationships instead of objects.

For example, suppose you want a column to show the quantity attribute of a relationship. Since this attribute could be on multiple relationship types, you should first add a column that contains the name of the relationship, using the expression `name` and applying it to relationships. Then, you could add a column with the expression `attribute[Quantity]` for the column's expression, also applying it to relationships.

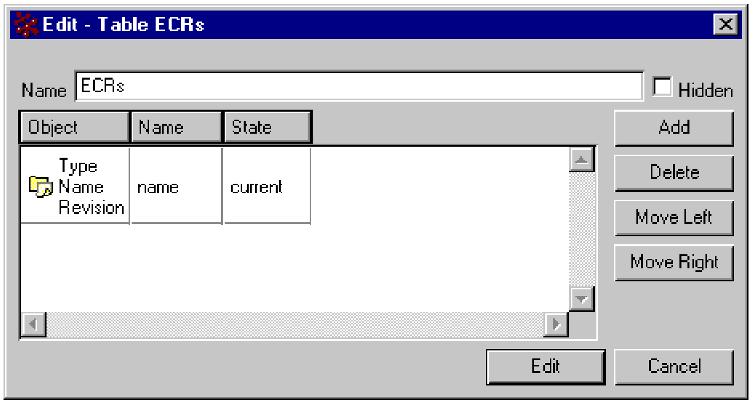
On the other hand, suppose you wanted to display the "as manufactured" information about multiple objects in one table, for use in a JSP table which does not display relationships. You could have a table with a column that showed the name of the objects connected with that relationship (with an expression defined as `from[BOM-As Manufact].to.name`), and another that showed the Quantity on the relationship (with an expression defined as `from[BOM-As Manufact].attribute[Quantity]`).

As you can see, expressions about relationships that apply to business objects are much more complicated than those that apply to relationships.

Editing a Table

To edit a table

1. Find the table object that you want to edit.
2. Double-click on it to open the Edit Table dialog.



Edit - Table ECRs

Name: ECRs

□ Hidden

Object	Name	State
Type Name Revision	name	current

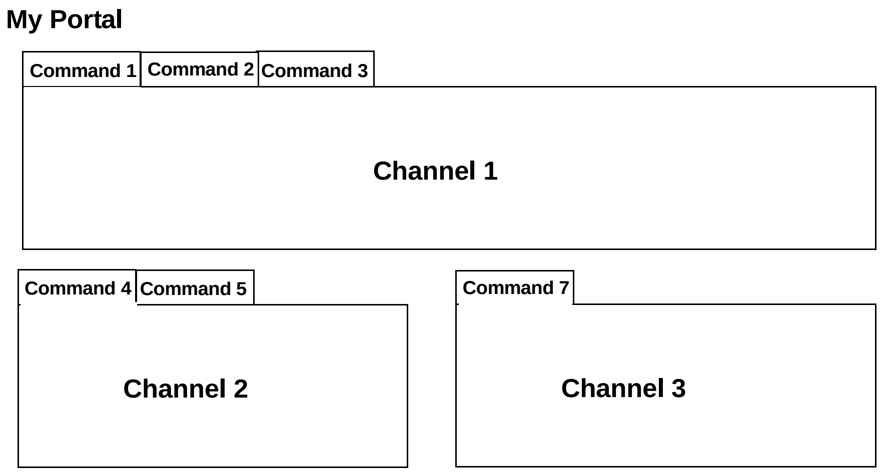
3. Work with the table as needed:
   - To add a column, click **Add**. Unlike the three buttons below it, the Add button is independent of any selection.
   - To delete a column, click the cell for the column underneath its heading. Do not click on the heading. Then click **Delete**. You cannot move or delete the first column, Object.
   - To move a column's location in the table, click the cell for the column underneath its heading. Then click **Move Left** or **Move Right** until the column is in the correct location.
   - To edit a column, double-click the column's heading.
4. When you are finished making your changes, click **Edit**.

14

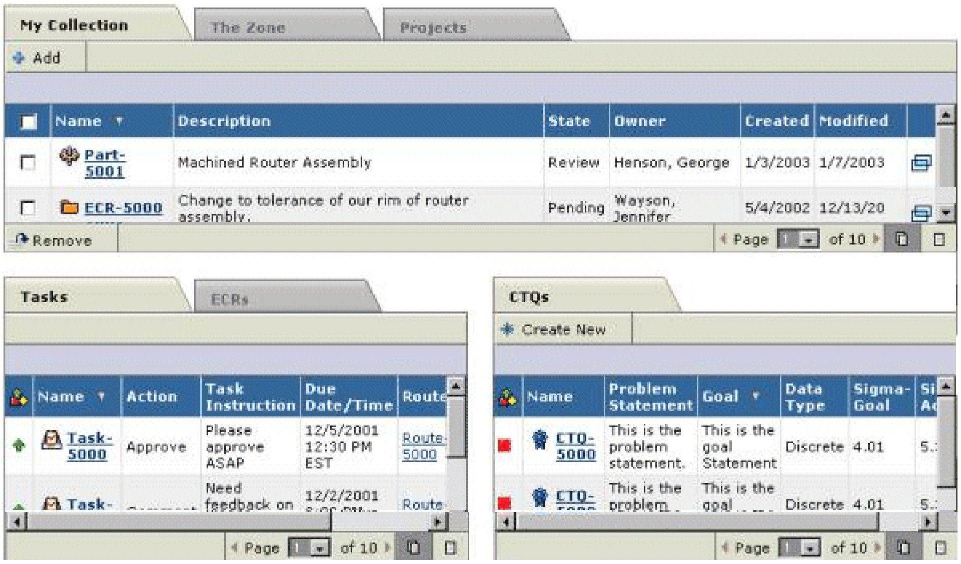
Working with Channels and Portals

Channels and Portals defined

Channels are essentially a collection of commands. They differ from menus in that they are not designed for use directly in an Toolkit application, but are used to define the contents of a *portal*, as detailed below.



A *Portal* is a collection of *channels*, as well as the information needed to place them on a Web page. Some channels and portals are installed with the Business Process Services and used in 3DEXPERIENCE products to display PowerView pages, but they can also be created for use in custom Java applications. The UIPortal bean (installed with the Business Process Services) is called to implement portals.



□	Name	Description	State	Owner	Created	Modified
□	Part-5001	Machined Router Assembly	Review	Henson, George	1/3/2003	1/7/2003
□	ECR-5000	Change to tolerance of our rim of router assembly.	Pending	Wayson, Jennifer	5/4/2002	12/13/20

Name	Action	Task Instruction	Due Date/Time	Route
Task-5000	Approve	Please approve ASAP	12/5/2001 12:30 PM EST	Route 5000
Task-		Need feedback on	12/2/2001	Route

	Name	Problem Statement	Goal	Data Type	Sigma-Goal	Si Ac
🟥	CTQ-5000	This is the problem statement.	This is the goal Statement	Discrete	4.01	5.:
🟥	CTQ-	This is the problem	This is the goal	Discrete	4.01	5.:

The use of the term "Portal" in this chapter refers to the Live Collaboration *administration object, and not to the broader internet definition described in JSR 168. In headings in the documentation we use the terms Live Collaboration portal and Public portal when a discernment needs to be made.*

Business Administrators can create channel and portal objects if they have the portal administrative access. Since commands are child objects of channels, commands are created first and then added to channel definitions, similar to the association between Live Collaboration types and attributes. Likewise, channels are created before portals and then added to portal objects. Changes made in any definition are instantly available to the applications that use it.

Defining a Channel Object

As a Business Administrator, you can create new channel objects if you have the portal administrative access. Channels are used in portal objects in a JSP application.

To define a channel object

1. Click ▨ or select **Channel** from the Object>New menu.

   The New Channel window displays, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Name

You must specify a unique name for each channel that you create. You should assign a name that has meaning to both you and the user. The name you choose is the name that will be referenced to include this channel within a portal.

Assigning an Icon

You can assign a special icon to the new channel. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

Type a description for the channel object. The description provides general information for you and the user about the function, use, or content of the channel. The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description. For example, if you were defining a channel named “Tasks” you might write a Description similar to one of the following. Each channel might differ considerably.

```
description "My Tasks"
description "Tasks for My Routes"
description "Incomplete Tasks"
```

Defining a Label

Type a label to appear in the application in which the channel’s portal is assigned. The label is displayed above the data in the channel.

Defining the Height

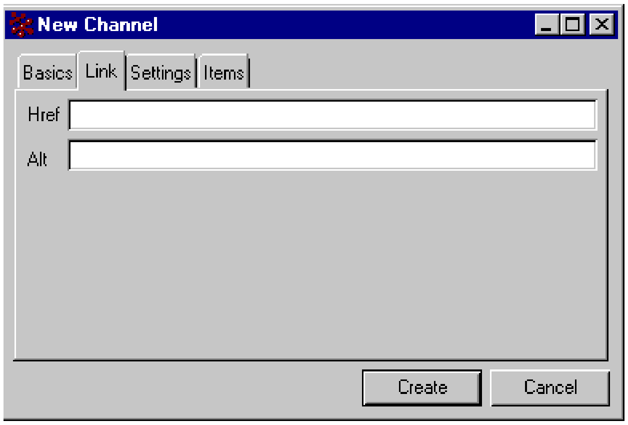
The height of the channel is used to indicate the height size in pixels the channel will occupy on the page. The default and minimum allowed is 260. If you enter a value less than 260, 260 will be used.

Defining the Hidden Option

Hidden objects are not displayed to Matrix Navigator users. Since there is currently no window that shows channels in Matrix Navigator, the hidden flag is not used at this time.

Defining the Link

Select the **Link** tab to provide information to be used when the channel is selected from an application page:



The information provided on the link tab is used to supply link data and alternative text to the href and alt HTML commands on the Web page that references the channel. Both fields are optional. Channels do not generally include link data.

Href

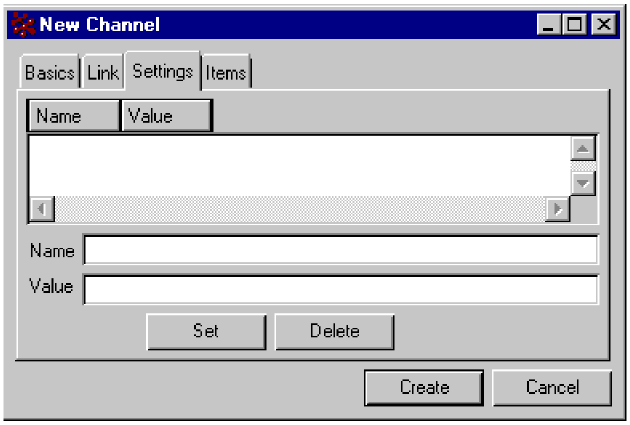
Type a string for the Href field to be used to provide link data to the JSP. The Href string generally includes a fully-qualified JSP filename and parameters.

Alt

Type any alternate text into the **Alt** field. This text is displayed until any image associated with the channel is displayed and also as “mouse over text.”

Defining the Settings

Select the **Settings** tab to provide any name/value pairs that the channel may need.

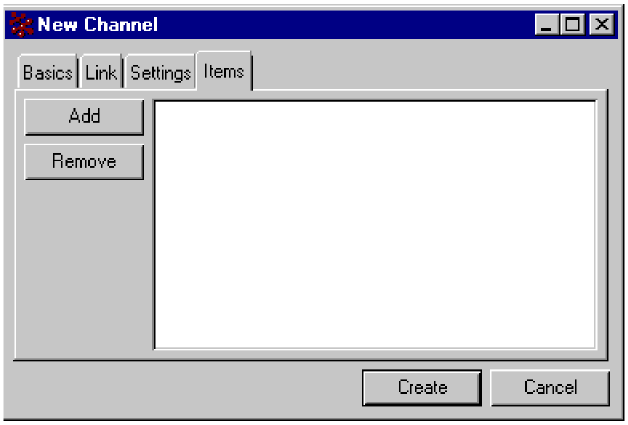


To add a setting, enter a **Name** and **Value** and click **Set**. You can remove a setting by selecting its name and clicking **Delete**.

Settings are general name/value pairs that can be added to a channel as necessary. They can be used by JSP code, but not by Hrefs on the Link tab. They are stored in the channel as an administrative property.

Assigning Channel Items

Select the **Items** tab to add commands to a channel.



Click **Add** to display a list of all existing commands. Select specific commands to be added to the channel you are creating. Each command will be displayed as a tab in the channel in the order in which it appears in the Items tab. You can drag commands to rearrange their order. Select a command and click **Remove** to delete it from the menu.

Creating the Channel

After you enter all appropriate information on the New Channel window, click **Create**. If you see an error message, correct the definition as required and click Create again.

Defining an Live Collaboration Portal Object

As a Business Administrator, you can create new portal objects if you have the portal administrative access. Portals can be used in custom Java applications. Before creating a portal, you must define the channels that it will contain.

To define a portal object

1. Click or select **Portal** from the Object>New menu.

   The New Portal window displays, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Name

You must specify a unique name for each portal that you create. You should assign a name that has meaning. The name you choose is the name that will be referenced to include this portal within an application.

Assigning an Icon

You can assign a special icon to the new channel. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

Type a description for the portal object. The description provides general information for you about the function, use, or content of the portal. The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description. For example, if you were defining a portal named "Management" you might write a Description clause similar to one of the following. Each portal might differ considerably.

```
description "Engineering Manager"
description "Program Manager"
description "Specification Manager"
```

Defining a Label

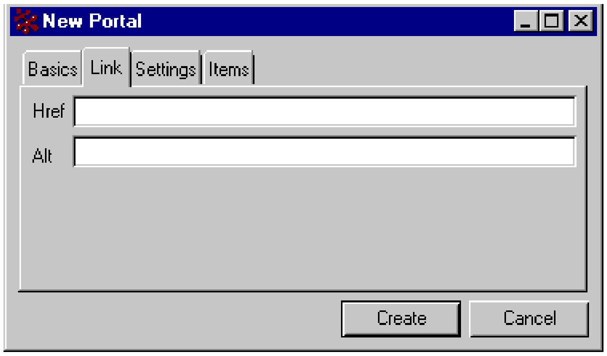
Type a label to appear in the application in which the portal is assigned. The label is displayed above the data in the portal.

Defining the Hidden Option

Hidden objects are not displayed to Matrix Navigator users. Since there is currently no window that shows portals in Matrix Navigator, the hidden flag is not used at this time.

Defining the Link

Select the **Link** tab to provide information to be used when the portal is selected from an application page:



The information provided on the link tab is used to supply link data and alternative text to the href and alt HTML commands on the Web page that references the portal. Portals do not generally include link data.

Href

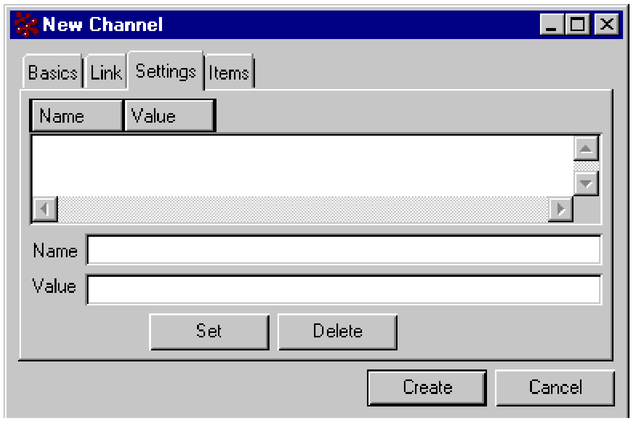
Type a string for the **Href** field to be used to provide link data to the JSP. The Href link is evaluated to bring up another page. The Href string generally includes a fully-qualified JSP filename.

Alt

Type any alternate text into the **Alt** field. This text is displayed until any image associated with the portal is displayed and also as “mouse over text.”

Defining the Settings

Select the **Settings** tab to provide any name/value pairs that the portal may need.



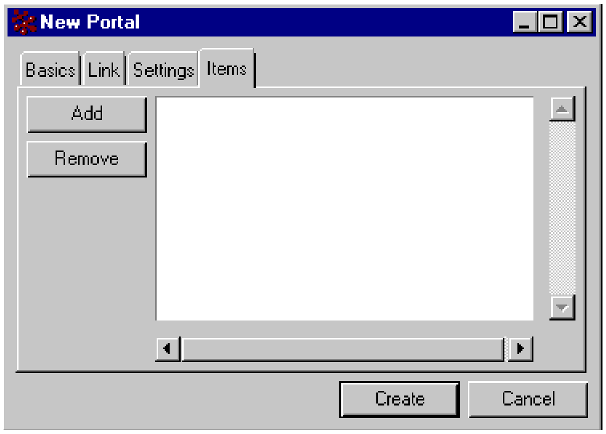
To add a setting, enter a **Name** and **Value** and click **Set**. You can remove a setting by selecting its name and clicking **Delete**.

Settings are general name/value pairs that can be added to a portal as necessary. They can be used by JSP code, but not by hrefs on the Link tab. They are stored in the portal as an administrative property.

Assigning Portal Items

Select the **Items** tab to add channels to the portal. You can use either of the following approaches to design your portal:

- Add items to the portal row by row.
- Add all items to the portal and create new rows with drag and drop.



To assign to and arrange items in a portal

1. Click **Add** to display a list of all existing channels.
2. Select the channels to be added to the portal you are creating and click **OK**. All the selected channels are shown in the first open row of the items pane.
3. Click **Add** to add more rows.
4. Drag and drop to rearrange as necessary. You can move items between or within rows.
5. To delete an item from the portal, select a channel and click **Remove**.

Creating the Portal

After you enter all appropriate information on the New Portal window, click **Create**. If you see an error message, correct the definition as required and click Create again.

15

Working With Web Forms

Web Forms Defined

As described in *Working With Commands and Menus*, administration objects can be defined that control the user interface (UI) of products, providing an application programming environment that allows solutions to be easily and consistently tailored to user requirements. This chapter provides information on how to create Web form objects in Business Modeler.

A *Web form* displays characteristics of a business object, including attributes, basics, or other properties. Each characteristic is listed in a row on the form page; each row on a page is a *field*. Web form pages can be read only, such as a page that shows details for an object, or writable, such as an edit page for an object.

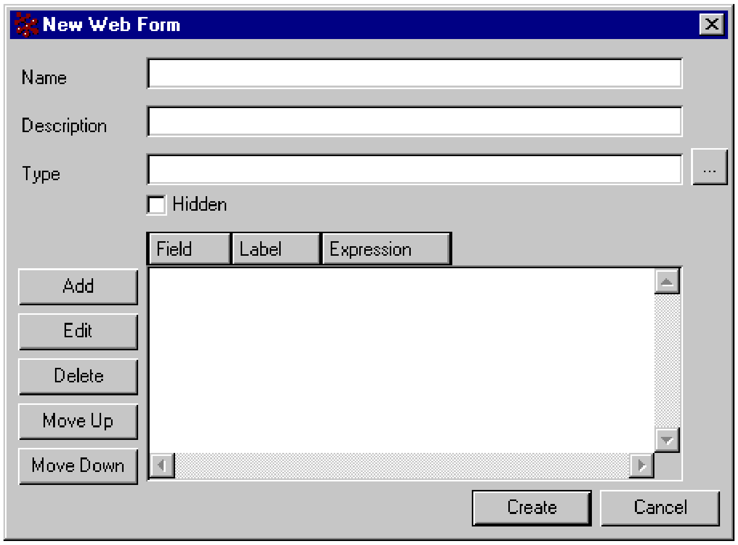
Defining a Web Form Object

As a Business Administrator, you can create new Web form objects if you have the Form administrative access. Web forms can be used in custom applications. Each field has several parameters where you can define the contents of the field, link data, user access, and other settings.

To define a Web form object

1. Click or select **Web Form** from the Object>New menu.

   The New Web Form dialog box opens.



2. Enter a name for the Web form in the **Name** text box.

   Web form names cannot include asterisks or commas. You must specify a unique name for each Web form you create. You should assign a name that has meaning to both you and the user. The name you choose is the name that will be referenced to use this table in an application.

3. For the **Description,** enter general information about the function of the form.

4. Indicate the object types with which the form is associated. A form does not have to be associated with a type.

   Type the type name in the **Type** text box.

   Or

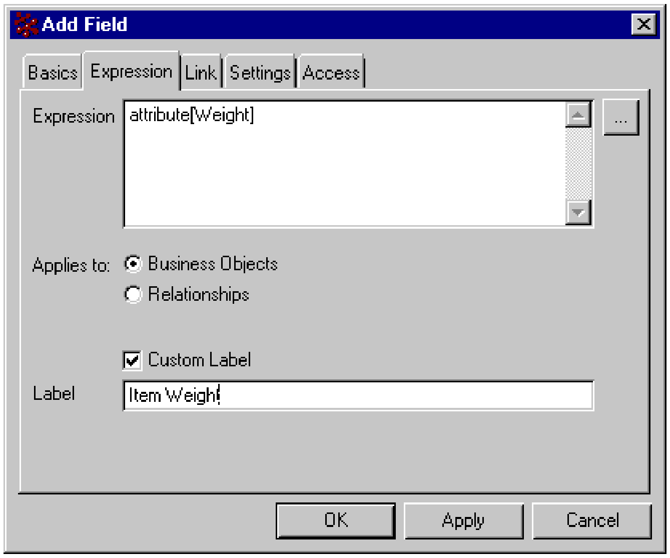
   Click the **Type** ellipsis button and select an object type from the Type Chooser that appears.

5. To make the Web form hidden, check **Hidden**.

   Hidden objects are not displayed to Matrix Navigator users. Since there is currently no window that shows business Web forms in Matrix Navigator, the hidden flag is not used at this time.

6. Click **Add** to add a field to the Web form.

   The Add Field dialog box displays, showing the **Expression** tab.



Expression: attribute[Weight]

Applies to: ● Business Objects ○ Relationships

☑ Custom Label

Label: Item Weight

7. Assign values to the field’s parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Field’s Expression

The **Expression** tab of the Add Field dialog box lets you define the data that should appear in the field.

Applying Expressions to Relationships vs. Business Objects

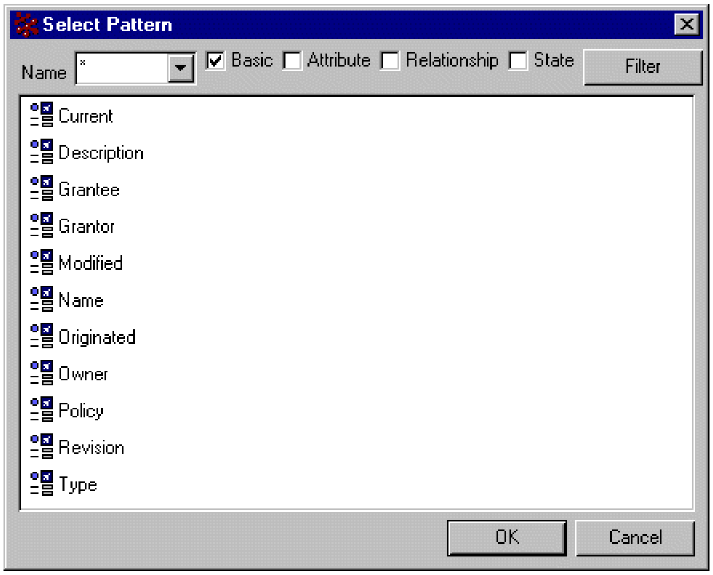
How you choose to apply a field expression depends on the information you want, how you define the expression, and how you want the information to display. You can get the same information into a form by defining a slightly different expression and applying it to relationships instead of objects.

For example, suppose you want a field to show the quantity attribute of a relationship. Since this attribute could be on multiple relationship types, you should first add a field that contains the name of the relationship, using the expression `name` and applying it to relationships. Then, you could add a field with the expression `attribute[Quantity]` for the field’s expression, also applying it to relationships.

To define a field’s expression

1. Click ... next to the Expression field.

   The Select Pattern chooser opens and lists the basic properties of an object. You can also display attributes and relationships by selecting the check boxes and clicking **Filter**. You can filter on the Name as well.



Name: *

☑ Basic ☐ Attribute ☐ Relationship ☐ State

2. Select the property you want to include as a form field and then click **OK**.

   The Expression tab shows the expression you just selected.

   You must define a new field for each MQL expression you want to use in the form. That is, you cannot use && (and) or || (or) qualifiers. However, you can use operators on numeric values. For example:

   ```
   attribute[Target Cost] - attribute[Actual Cost]
   ```

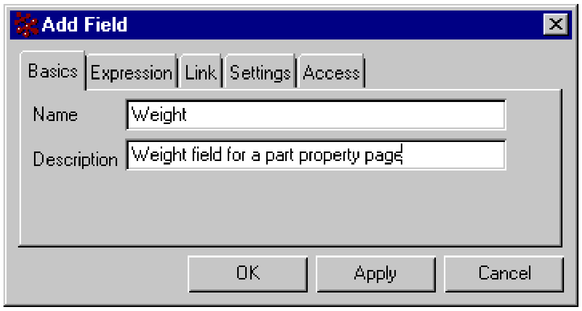
   For help formatting expressions, *Configuration Guide : About Selectables*.

3. Select whether the expression applies to **Business Objects** or **Relationships**.

4. To enter a unique name for the field label, check **Custom Label** and enter the name in the **Label** text box. If you uncheck Custom Label, the field label defaults to the expression of the field.

Defining a Field's Basics

Click the **Basics** tab to specify a unique name and description for each field you create. Enter a **Name** for the field. The name you choose is the name that will be referenced when specifying this field in an application.



Name: Weight

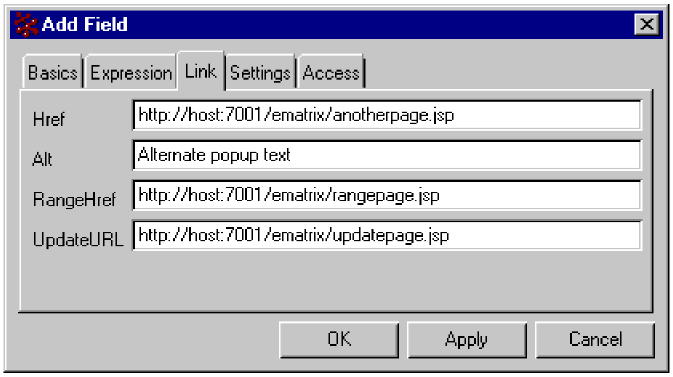
Description: Weight field for a part property page

Add a **Description** for the field. The description provides general information for you about the function, use, or content of the field. The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description.

Defining the Field’s Link

Click the **Link** tab to specify information about the Href URL to go to when a user clicks the field’s data.

:



Href: http://host:7001/ematrix/anotherpage.jsp

Alt: Alternate popup text

RangeHref: http://host:7001/ematrix/rangepage.jsp

UpdateURL: http://host:7001/ematrix/updatepage.jsp

Href

Enter the URL that should be called when a user clicks the data in the field. For example, for a Web form page that represents the details page for part business objects, an ECR field might contain the name of a connected ECR and users can click the name to go to a details page for the ECR. Many Web form fields will not have an Href value because the field’s data is not clickable. The Href string generally includes a fully qualified JSP filename and parameters, which can contain embedded macros and expressions for mapping to database schema. Refer to *Using Macros and Expressions in Configurable Components* for more details.

Assigning an href to a link can create problems if a user clicks the same link twice when initiating such actions as changing an object’s state or submitting a form. The reason for this is as follows. An href assigned to a link is considered a server request, even if it is a JavaScript command. Whenever the browser detects a server request, the browser stops processing the current request and sends the new request. Therefore, when a user first clicks on an href link, the request is processed, and, typically, a JSP page starts executing. If, during this time, a user clicks the same link again, the first request is interrupted before completion and the new request is processed instead. To avoid this scenario, you can set the href to “#” and use the onclick event instead. The generic code for this is:
`<a href="#" onclick="submitForm()">`.

Alt

Type any alternate text into the **Alt** field. This text is displayed until any image associated with the field is displayed and also as “mouse over”, ToolTip text.

RangeHref

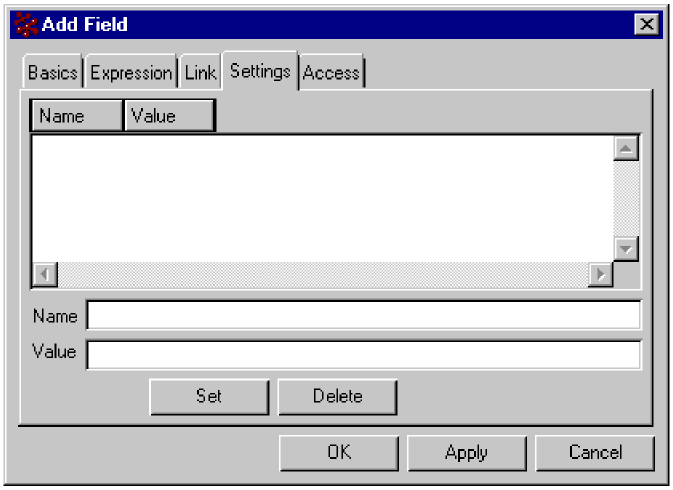
Specifies the JSP that gets a range of values and populates the field with the selected value. These values can be displayed in a popup window or a combo box.

UpdateURL

Specifies the URL page that should be displayed after the field is updated.

Defining Settings for a Field

Choose the **Settings** tab to provide any name/value pairs that the field needs.

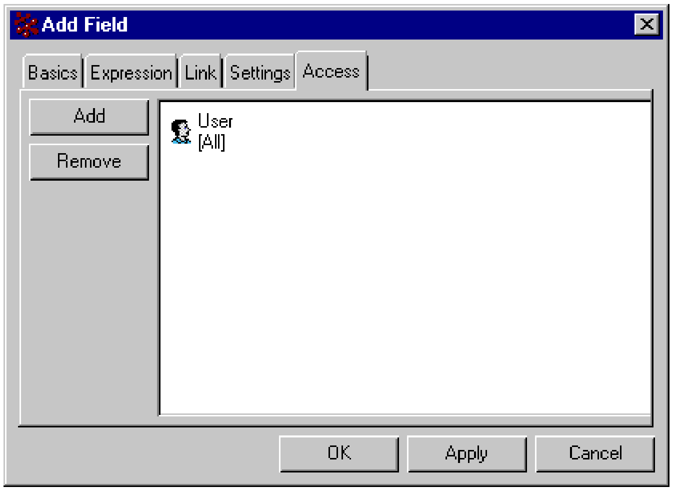


To add a setting, enter a **Name** and **Value** and click **Set**. You can remove a setting by selecting its name and clicking **Delete**.

Settings are general name/value pairs that can be added to a field as necessary. They can be used by JSP code, but not by hrefs on the Link tab. Also refer to *Using Macros and Expressions in Configurable Components* for more details.

Defining Access to a Field

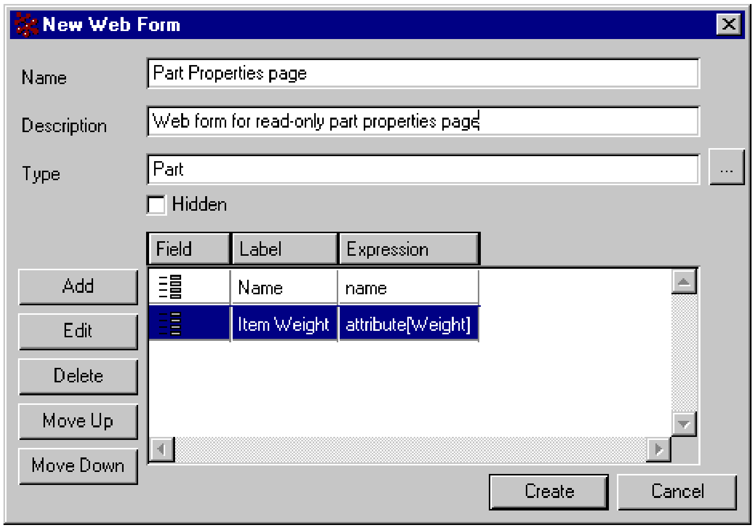
Choose the **Access** tab to define the users allowed to see the field.



Any number of roles, groups, persons, and/or associations can be added to the field (role-based in this case includes all types of users and is not limited to only Live Collaboration roles). User [All] is specified by default. Click **Add** and a list of all existing (and not hidden) users is displayed. Limit access by selecting those specific users who will be allowed access to the field and click **OK**. Select a user and click **Remove** to remove a user's access to the field.

Creating the Field and Web Form

After you enter all appropriate information on the Add Field dialog box, click **OK**. The field is added to the Web form. Repeat the process for all other fields.



Name: Part Properties page

Description: Web form for read-only part properties page

Type: Part

□ Hidden

Field	Label	Expression
	Name	name
	Item Weight	attribute[Weight]

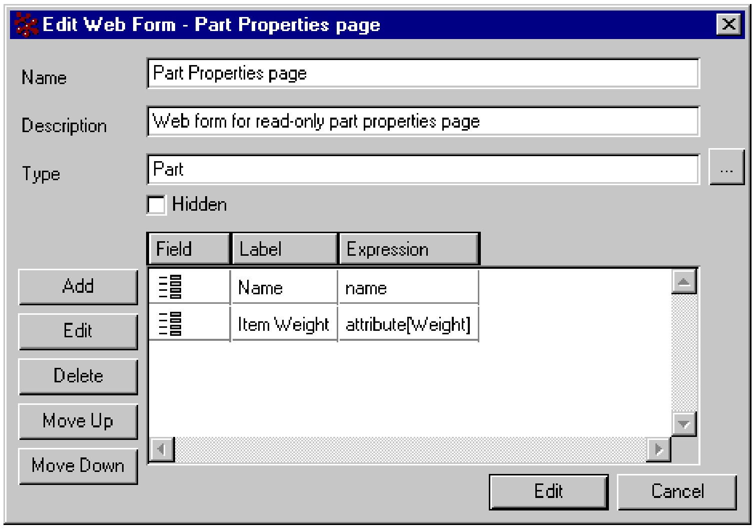
You can rearrange and delete fields by selecting a field (not in the heading, but in the area that displays the expression) and clicking **Move Up**, **Move Down**, and **Delete**. If you need to edit a field, select it and click **Edit**. When you are satisfied with the Web form, click **Create**.

Editing a Web Form

To edit a Web form

1. Find the Web form object that you want to edit.

2. Double-click on it to open the Edit Web Form dialog.



Edit Web Form - Part Properties page

Name	Part Properties page
Description	Web form for read-only part properties page
Type	Part

□ Hidden

Field	Label	Expression
☰	Name	name
☰	Item Weight	attribute[Weight]

3. Work with the form as needed:

   - To add a field, click **Add**. Unlike the four buttons below it, the Add button is independent of any selection.
   - To edit a field, select the field by clicking on it and then click **Edit**.
   - To delete a column, select the field and then click **Delete**.
   - To move a field’s location in the table, select it and hen click **Move Up** or **Move Down** until the column is in the correct location.

4. When you are finished making your changes, click **Edit**.

16

Working With Applications

Overview

You can design your company’s data model such that you can mark part or all of it as private or protected. The data you mark private cannot be accessed from any other project while the data you mark protected can only be viewed and not modified. If your company is working on a top secret government project, you will need to mark the data connected to this project as private to avoid any accidental viewing or modification of data. You can do this by assigning an owning application to your project.

An *application* is a collection of administrative objects (attributes, relationships and types) defined by the Business Administrator and assigned to a project. It acts as a place where all application dependent associations to these other administration objects are defined. The application members (the types, relationships etc) can have different levels of protection (private, protected or public). This protection also extends to the objects governed by them. For example, if you mark a relationship as protected, all connections of that type will also get marked as protected.

The advantage of assigning an owning application to a project is that it ensures data integrity. All modifications to the data are handled by the owning application code which performs all necessary checks to ensure complete data integrity. This prevents any unintended mishandling and possible corruption of data. Thus, when an 3DSEXPERIENCE Platform product (or any custom application) tries to access data marked private or protected by connecting to the Live Collaboration Server, it has to specify the name of the owning application containing that data. A placeholder for the application name is added to the Studio Customization Toolkit Context object allowing the passing of this information when a user is authenticated.

You can specify an application name only in the Studio Customization Toolkit.

In MQL and Business Modeler, a person can be assigned an owning application. This default application is used in place of the application name passed during Studio Customization Toolkit login allowing Studio Modeling Platform users to access private or protected data.

Defining an Application

There are several parameters that can be associated with an application. Each parameter enables you to provide information about the new application. While only a name is required, the other parameters further define the application, as well as provide useful information about the it.

To define an application

1. Click or select **Application** from the Object>New menu.

   The New Application dialog box opens, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining the Name

You can specify the name of the application you are creating. All applications must have a unique type name assigned. For additional information, refer to *Administrative Object Names* in Chapter 1.

The application name will appear whenever the application is listed in an Live Collaboration window.

Defining a Description

A description provides general information about the application. The description can consist of a comment or qualifying phrase.

```
For CATIA models

For DELMIA models
```

Although you are not required to provide a description, this information is helpful when a choice is requested.

Defining the Hidden Option

You can specify that the new application is “hidden” so that it does not appear in the Application chooser or in any dialogs that list applications in Live Collaboration.

Hidden objects are always accessible through MQL.

Selecting Application Members

You can assign members to an application. Application members can be an Attributes, Relationships or Types that are already defined within the database. To each application member you can also assign a level of protection.

By default the level of protection is set to public.

A user who does not have access to a protected type still has toconnect and todisconnect access on objects of this type. However, fromconnect and fromdisconnect access are not allowed.

To select a member

1. From the Members Tab of the New Application dialog, click Add.

   The Member Chooser opens.

2. Select one or more Attributes, Relationships or Type in the dialog box. Use Ctrl-click to select multiple members.

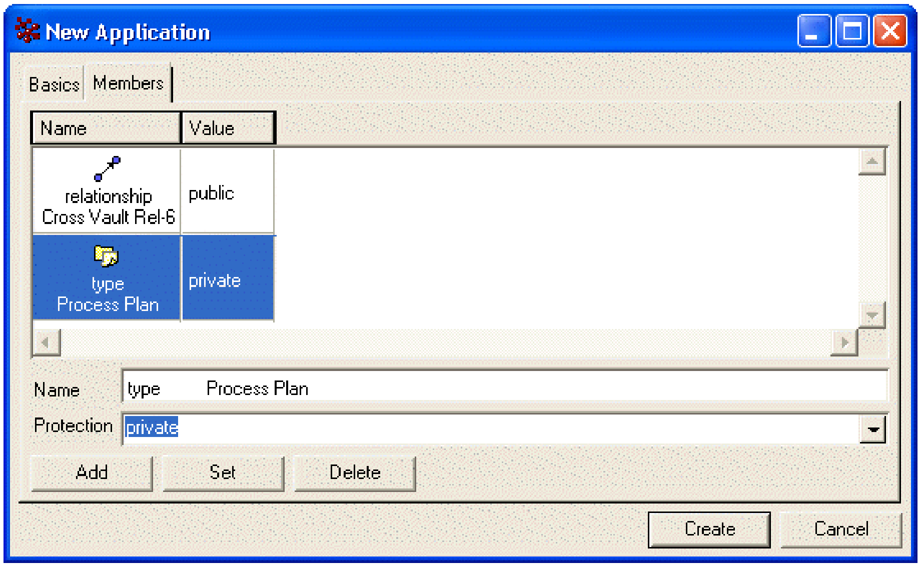
3. Select the level of protection that you want the application member to have from the pull down menu.

   Application members can have the following levels of protection:

   - private
   - protected
   - public

4. Click **Ok**.

The selected members are listed in the Members table.



Name	Value
relationship Cross Vault Rel-6	public
type Process Plan	private

Name: type Process Plan

Protection: private

Creating the Type

After you enter all appropriate information in the New Application dialog box, click Create. The new application object is created.

Modifying an application

After an application is defined, you can modify the application. Refer to the description *Modifying an Administrative Object*.

17

Working With Relationship Browsers

Overview

In an object-oriented database, administrative objects are modular by design; simpler pieces comprise more complex components. For example, when you create a type, you assign it attributes. Likewise the type can be associated with several policies. If changes need to be made to the database definitions, it is helpful to look at the existing relationships between administrative objects to assess the impact of the change. These connections can be displayed graphically using the *Star* or *Indented browsers*.

The Star browser shows related objects clustered around the selected object, and has arrows indicating that a relationship exists. It can be displayed in either a circular or spiral format.

The Indented browser shows a hierarchy of related objects on indented lines beneath the selected object. A scroll bar allows you to see additional “screens” of information.

In each view of related objects, you can apply filters to control the type of relationship explored and the type of object you see in your browser.

Star Browser

The Star browser shows related objects clustered around the selected object, and has arrows indicating that a relationship exists.

You can choose to display the objects in a circular or spiral pattern.

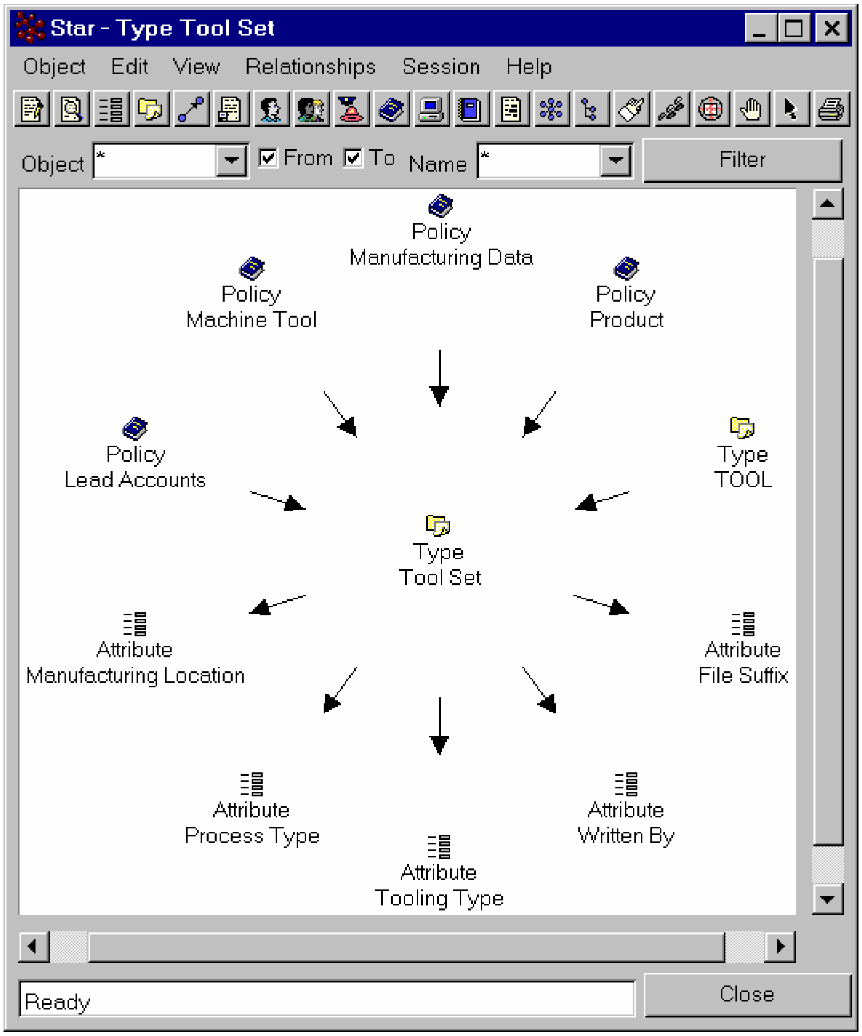
- The Star browser in the circular format is limited to the number of objects that will physically fit around the selected object displayed on your screen. To display all related objects when the limit is reached, use the spiral format which will spiral the display off the screen and provide scroll bars, as necessary.
- The Star browser in the spiral pattern will not look significantly different from the circular pattern when only a few objects are displayed. However, when a large number of objects are connected, the spiral pattern will prevent the display of overlapping objects and increase readability.

In the Star browser, you can apply filters to control the type of objects you see in your browser.

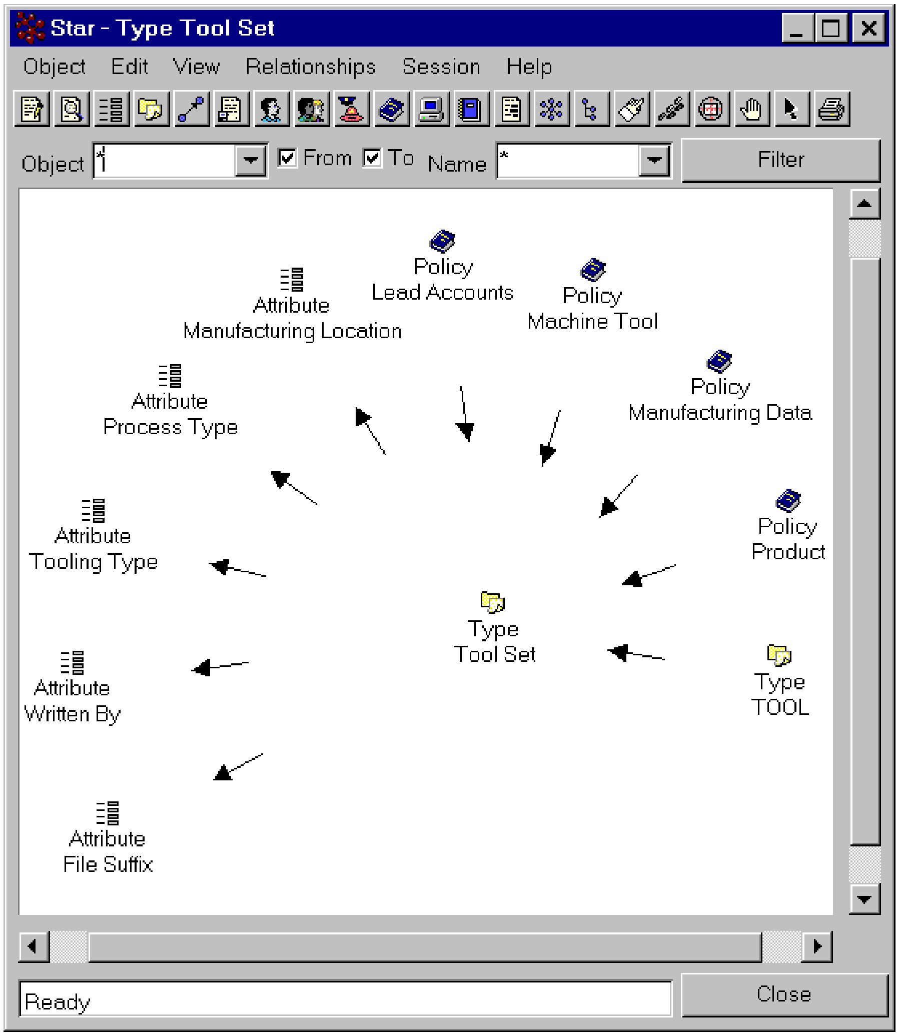
To activate the Star browser

1. Select the object you want to explore.
2. Click or select **Star** from the Relationships menu or click the right mouse button on the object and choose **Star** from the popup menu.

   The Star browser appears displaying related objects clustered around the selected object.



3. Select **Spiral** from the View menu to display the spiral pattern, as shown below.



4. Select **Circle** from the View menu to return to the circular pattern.

Navigating with the Star Browser

Once the Star browser opens, you can navigate through database relationships in three ways:

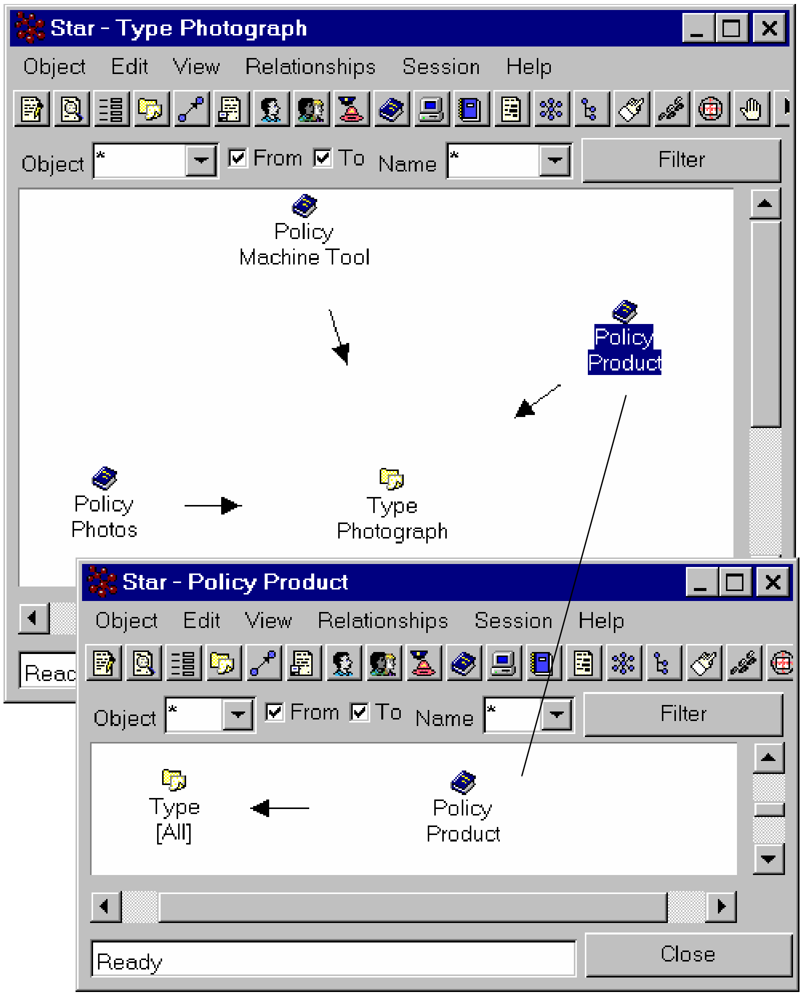
- By displaying a Star browser of a connected object.
- By using the There and Back tools.
- By displaying an Indented browser of a connected object.

To navigate within the Star browser

1. Select the object you want to explore in the Star browser.

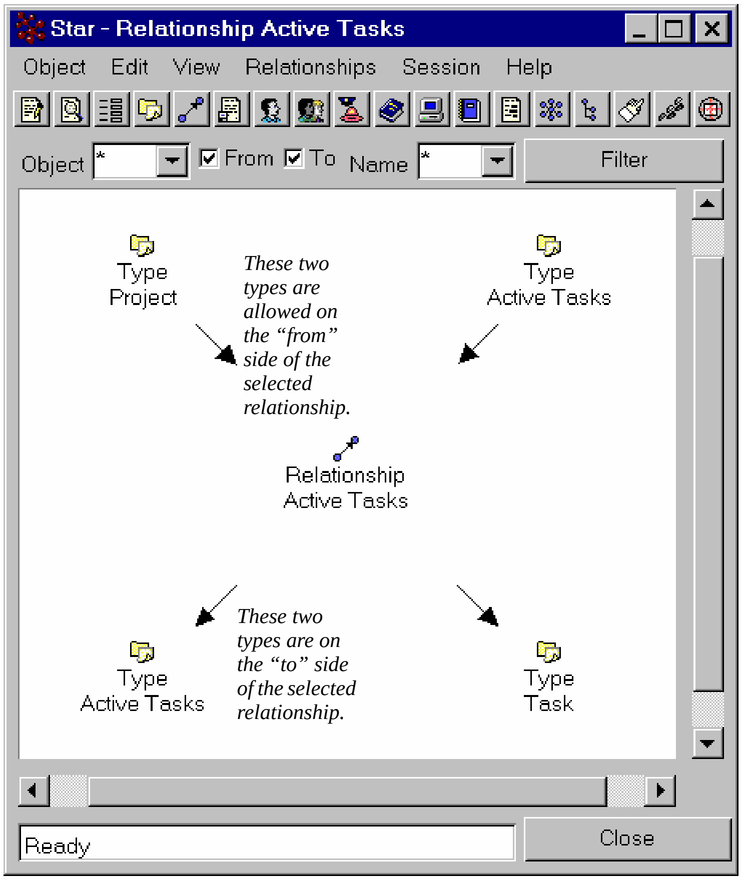
2. Click ❋ or select **Star** from the Relationships menu or click the right mouse button on the object and choose **Star** from the popup menu.

   Another Star browser opens for the selected object. In the example below, Policy Product is selected, and the resulting browser is shown.



Reviewing Relationships

Notice that some arrows on the Relationship Browser point “to” the selected object while others point “from” the object. The object on the flat side of the arrow is referred to as the “from” object, while the object near the arrowhead is known as the “to” object.



These two types are allowed on the “from” side of the selected relationship.

These two types are on the “to” side of the selected relationship.

You can filter based on the object type (or pattern) you want to show or based on the object name. See *Using Filters on the Browsers* for examples.

Business Browser Relationships

The following table shows the From and To Relationships for each business object type:

Business Object Type	**Relations (From)** ←	**Relations (To)** →
Association	none	none
Attribute	Type Relationship	none
Form	none	Type
Format	Policy	none
Group	Parent group	Child group Person
Person	Group Role	none
Policy	none	Format Store Type

Business Object Type	Relations (From) ←	Relations (To) →
Program	Format Type Attribute Relationship Policy Wizard	none
Relationship	From Type	To Type Attribute
Role	Parent role	Child role Person
Type	Policy Relationship Parent Type Form	Attribute Method (Program/Wizard)
Wizard	Program	none

There and Back

When you select either the Star browser or the Indented browser, the There and Back Tools are available from the toolbar or Relationships menu. These tools enable you to jump to the Star browser display of another object *without creating a new browser*. Then, you can jump back to the previous Star browser display. This is different than displaying additional Star browsers because you will have only one browser to close when using There and Back.

To navigate with There and Back

1. Select an object from the Star or Indented browser.
2. Select the tool.

   Or

   Select **There** from the Relationships menu. The browser displays changes from the current object to the newly selected object.
3. To return to the original display, select the tool.

   Or

   Select **Back** from the Relationships menu.

   All jumps are remembered and you can continue to go back through history with this tool.

The Indented browser is also available from the Star browser. Refer to the *Indented Browser* section for more information.

Additional Tools

The Star browser contains many of the same menus and tools as the Primary browser. This enables you to select any menu or tool and perform any of the existing operations on the selected object without changing from the Star browser. Refer to *Business Modeler Primary Browser: Menus, Options, and Tools* in Chapter 1 for a review of these tools.

When you select the Star browser, these additional tools are available:

✋	The *Positioning tool* enables you to randomly move the Star browser for easier viewing. This feature is useful when the objects will not all fit in the browser. The screen cursor changes from an arrow to a hand. Drag the hand cursor to position the browser. The browser moves as though you were pushing a piece of paper around on a tabletop.
↖	The *Selecting tool* resets the viewing mode from random movement (if the Positioning tool was clicked) to the standard point-and-select mode.
⊕	The *Centering tool* centers the relationship structure on your screen. When you select this tool, the browser is centered automatically.

Indented Browser

The Indented browser shows a hierarchy of related objects on indented lines beneath the selected object. When you view related objects in an Indented browser, you can apply filters to control the type of objects you see in your browser. Filters are described in *Using Filters on the Browsers*.

Once the browser opens on your screen, you can use it to navigate through database relationships, as you will see in the following sections.

To activate the Indented browser

1. Select the object you want to explore.
2. Activate the Indented browser in any of three ways:

   Select the tool.

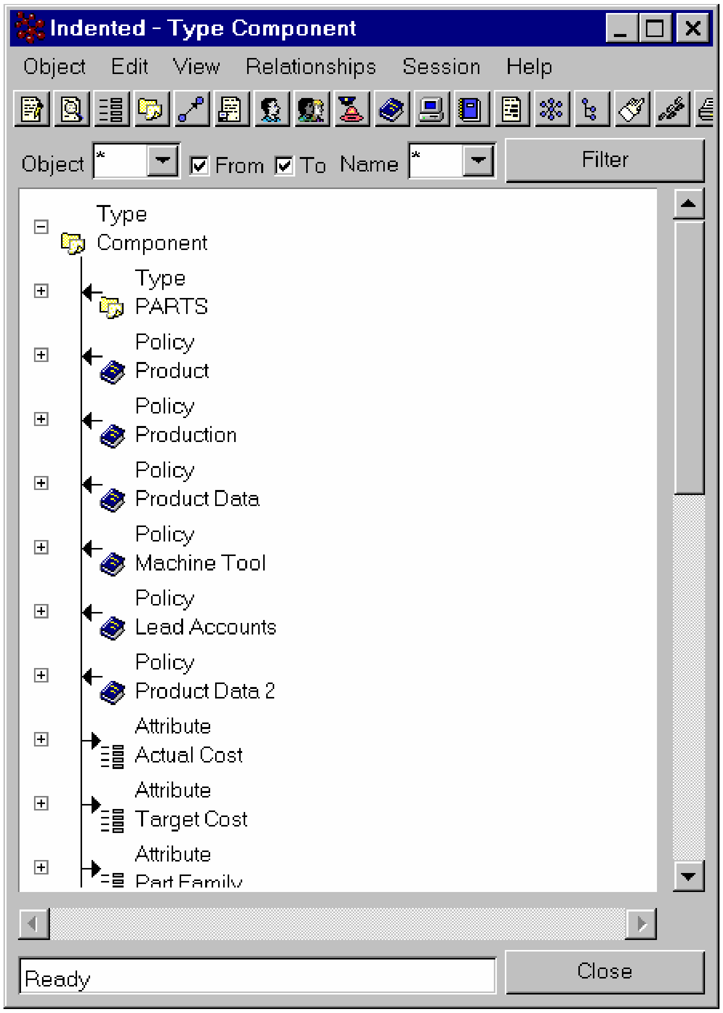
   Or

   Select **Indented** from the Relationships menu.

   Or

   Click the right mouse button on the object and choose **Indented** from the popup menu.

   The Indented browser appears displaying related objects on indented lines beneath the selected object:



The Indented browser contains many of the same menus and tools as the Primary Browser, enabling you to perform any of these operations on a displayed object without changing from the Indented browser.

Navigating With the Indented Browser

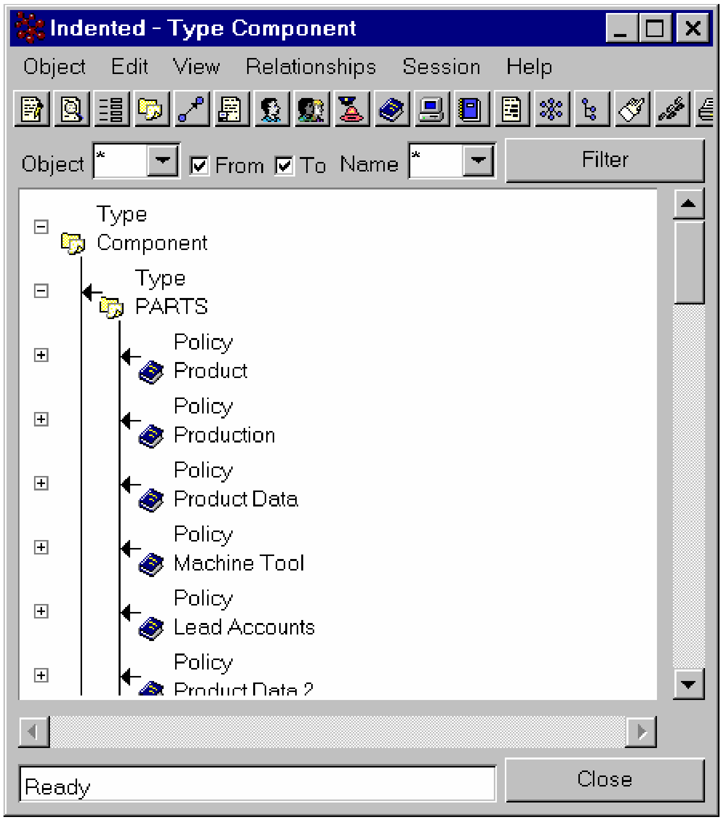
Once the Indented browser opens, you can navigate through database relationships in four ways:

- By displaying an Indented browser of a connected object, as outlined in *Indented Browser*.
- By displaying a Star browser of a connected object, described in *Star Browser*.
- By using the There and Back tools, as described in *To navigate with There and Back*.
- By Expanding a connected object. This is described in the section that follows.

Expanding the Indented Browser

You can expand the Indented browser in three ways: one level, multiple levels, or the entire multi-level tree. When you first activate the Indented browser, one level is displayed. The (+) symbol (the **Expand** button) to the left of a displayed object enables you to expand the hierarchical display one level.

For example, if you click the **Expand** button to the left of Type PARTS, another level of detail is provided.



Object * ☑ From ☑ To Name *

Type Component
Type PARTS
Policy Product
Policy Production
Policy Product Data
Policy Machine Tool
Policy Lead Accounts
Policy Product Data 2

Ready

You can click other **Expand** buttons to display additional levels of detail. In addition, you can quickly display the entire tree by holding down the Shift key when you click the **Expand** button.

When you click the **Expand** button, the symbol changes to (-) (the **Collapse** button). Click the **Collapse** button to *collapse* or remove a level of the display.

Additional Tools

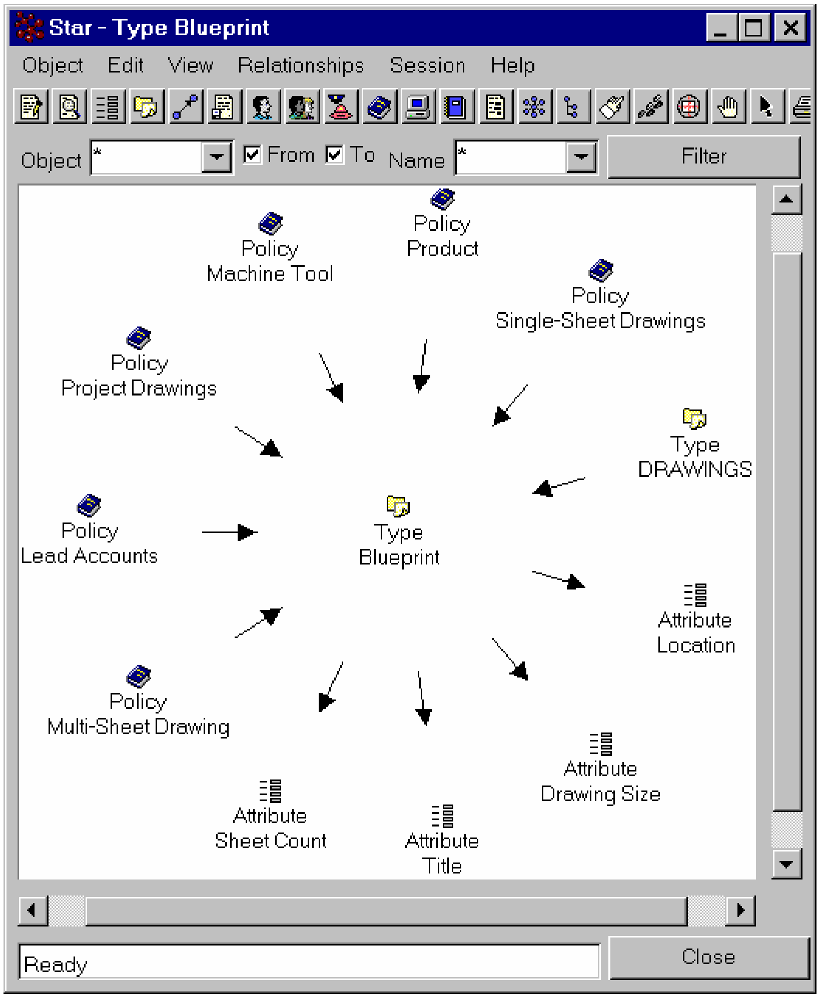
The Indented browser contains many of the same menus and tools as the Primary Browser. This enables you to select any menu or tool and perform any of the existing operations on the selected object without changing from the Indented browser. Refer to *Business Modeler Primary Browser: Menus, Options, and Tools* in Chapter 1 for a review of these tools.

When you select the Indented browser, these additional tools are available:

✋	The *Positioning tool* enables you to randomly move the Star browser for easier viewing. This feature is useful when the objects will not all fit in the browser. The screen cursor changes from an arrow to a hand. Drag the hand cursor to position the browser. The browser moves as though you were pushing a piece of paper around on a tabletop.
↖	The *Selecting tool* resets the viewing mode from random movement (if the Positioning tool was clicked) to the standard point-and-select mode.

Using Filters on the Browsers

The Filter bar allows you to view specific relationships of an object. For example, if you open a Star or Indented browser on a type named Blueprint, you see all objects related to Blueprint. The objects will include policies, types, and attributes.



If you want to view, for example, only the *policies* related to the type Blueprint, you can filter out the other objects.

The Filter Bar contains text boxes with a drop-down list which indicate the elements on which you can filter. The names in the list vary depending on the type of object displayed. Each element provides a way for you to filter the information displayed in the work area.

Two filters are available in both of the browsers for all object types:

- **Object**Specifies the related administrative object types.
- **Name**Specifies the name of the object(s).

The default value for each filter text box is always the wildcard (*), meaning all. To show only the items that meet some criteria, enter the desired values in the filter text boxes and click the **Filter** button.

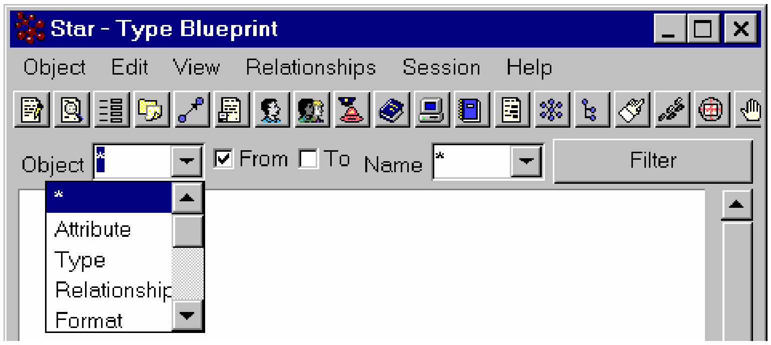
To show all objects again, simply enter * in each text box and click the **Filter** button.

To filter in a browser

1. Type the name of an object in the **Object** text box.

   Or

   Click the down arrow on the **Object** text box and select an object from the list.



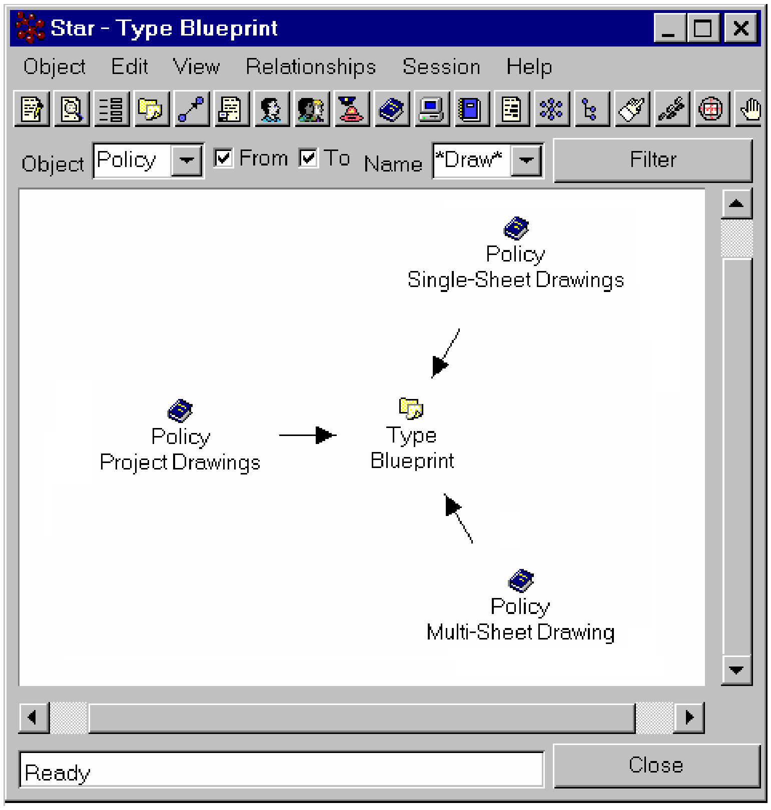
2. Check the **From** check box to display objects that are connected *from* the selected object.
3. Check the **To** check box to display objects that are connected *to* the selected object.
4. In the **Name** text box, type the name, or type part of the name using wildcards. End your input with an asterisk to find any objects that start with the letters you type. Start your input with an asterisk to find any objects that end with the letters you type.

   For example, to find objects whose names start with either Project or Product, you could type **Pro*** in the Name text box. To find objects whose names end with Drawings (e.g., Product Drawings, Project Drawings, etc.), you could type ***Drawings** in the Name text box.

   If you type asterisks both at the beginning and end of the text, Live Collaboration finds any object with names that contain these letters. For example, if some objects have been named “Drawings” and other have been named “Drawing,” use ***Draw*** to find all these objects.

Live Collaboration names are case-sensitive.

5. Click **Filter** to display the filtered objects.



Object Policy ☑ From ☑ To Name *Draw*

Ready

Printing the Contents of the Browser

Star and indented browsers have a print tool in the tool bar. To print the contents of the browser, click 🖨. The standard Windows Print dialog box opens. Make your selections and click **OK**.

18

Working With Pages

Overview

A *page* is a type of Live Collaboration administrative object that is used to create and manage properties for Java applications. Live Collaboration first looks for a property file using the classpath, but if the file is not found, it looks for a page object of the same name. In a distributed environment, such as a RMI gateway configuration, this allows you to centralize all property files and propagate updates immediately.

In environments that use both the Studio Modeling Platform and 3DEXPERIENCE Platform products, when emxSystem.properties and emxFrameworkStringResource.properties (including translated versions of the later) are stored in the database, certain errors can be avoided.

You can create, modify, and delete pages using the Business Modeler application. You can also use standard MQL commands such as create, delete, copy, modify, print, and list to manage page objects.

Page objects currently are not supported in a J2EE environment.

Defining a Page Object

As a Business Administrator, you can create new page objects. You can copy the contents of any property file into the Content tab of the New Page dialog (or type in the settings manually). You must then save the page with the same name as the original property file.

To define a page object

1. Click 🌐 or select **Page** from the Object>New menu.

   The New Page dialog box displays, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Name

You must specify a unique name for each page that you create. You should assign a name that has meaning to both you and the user. The name you choose is the name that will be referenced to include use this page by a Web page. The page name is limited to 127 characters.

Assigning an Icon

You can assign a special icon to the new page. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

Type a description for the page object. The description provides general information for you and the user about the function, use, or content of the page.

The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description.

Specifying the MIME TYPE

You can associate a MIME (Multi-Purpose Internet Mail Extension) type with a page. It defines the content type of the file.

The format of MIME types is a type and subtype separated by a slash, for example, text/plain or text/jsp.

The major MIME types are application, audio, image, text, and video. There are a variety of formats that use the application type. For example, application/x-pdf refers to Adobe Acrobat Portable Document Format files. For information on specific MIME types (or more appropriately-called “media” types) refer the Internet Assigned Numbers Authority Web site at http://www.isi.edu/in-notes/iana/assignments/media-types/. The IANA is the repository for assigned IP addresses, domain names, protocol numbers, and has also become the registry for a number of Web-related resources including media types.

This field is not required for use with properties files.

Date and Time

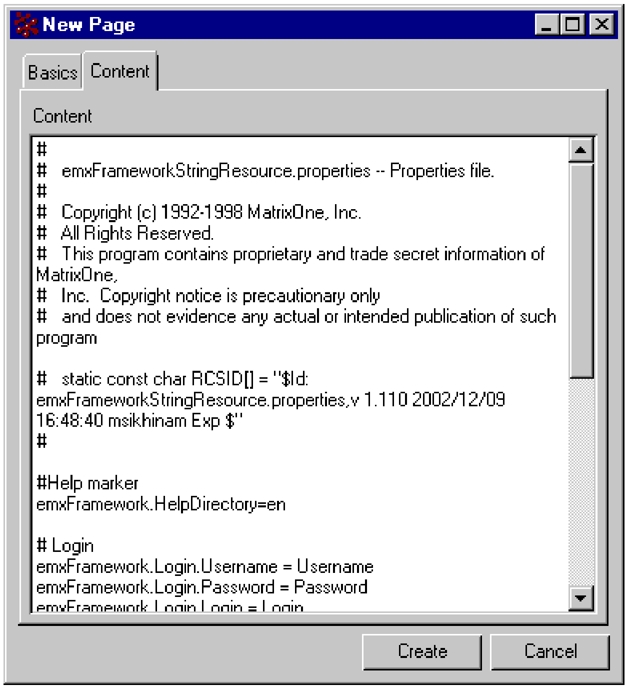
The system automatically includes the date and time that the page is originally created, and the date and time of the last modification.

Defining the Hidden Option

You can mark the new page as “hidden” so that it does not appear in the chooser in Live Collaboration. Users who are aware of the hidden page’s existence can enter its name manually where appropriate. Hidden objects are also accessible through MQL.

Defining Page Content

Select the **Content** tab to display an input area where you can include the page code:



```
#
#   emxFrameworkStringResource.properties -- Properties file.
#
#   Copyright (c) 1992-1998 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:
emxFrameworkStringResource.properties,v 1.110 2002/12/09
16:48:40 msikhinam Exp $"
#

#Help marker
emxFramework.HelpDirectory=en

# Login
emxFramework.Login.Username = Username
emxFramework.Login.Password = Password
emxFramework.Login.Login = Login
```

This Content page is a built-in text editor, where you can make changes to the settings without need of another editor. However, you can do your editing in any editor of your choice, then copy and paste the code into this window, or save it as a file and use MQL to put the file text into the content of the page.

Supporting Alternate Languages and Display Formats

For global database access, pages generally need to be provided in multiple languages. And with the wide use of cell phones and other hand-held devices in accessing Web pages, you may also need to support the page's display on a small LCD in wireless markup language (wml). When this is the case, you can use the following Studio Customization Toolkit call to open a page with a language and format argument, following the syntax:

```
open(BASE_PAGE_NAME, LANG, MIMETYPE)
```

For example:

```
open(login.jsp, fr, wml)
```

When evaluating this code, the system first looks for the file named `login_fr_wml.jsp`. If this page is not found, it then attempts to find `login_wml.jsp`. As a last resort, it searches for the page `login.jsp`. Of course, you could call `login_fr_wml.jsp` directly, but the addition of arguments gives you much more flexibility when writing the code.

In this case, you would first create `login.jsp`. Next, if you wanted to support wml, you would then create the wireless version of the page and name it `login_wml.jsp`. Then for each language you want to support, you would translate the text portions of the page(types) and save as `login_LANG.jsp` and `login_LANG_wml.jsp`. For example, to support multiple languages you might have pages with the following names in the database:

```
login.jsp
login_ch-tw.jsp
login_ch-gb.jsp
login_it.jsp
```

To then add support for wml for these languages, you might add the following pages:

```
login_wml.jsp
login_ch-tw_wml.jsp
login_ch-gb_wml.jsp
login_it_wml.jsp
```

Note that the base page does not have to be in English. Also, the LANG argument could be more than two characters, such as en-us, en-uk, or ch-tw.

19

Working With Resources

Overview

A *resource* is a Live Collaboration administrative object that stores binary files of any type and size. Applications use resources to display output to a standard Web browser or a small LCD device by providing components for Web pages. They are often .gif images, but they could be movie or video files or any resource that you use in a Web application, including:

- GIF
- JPEG
- MPEG
- AVI
- WAV
- JAR
- CAB

For example, you can include in the database a resource that represents the company corporate logo. In a Web application, the company corporate logo may be referred to many times.

The resource editor allows you to name the administrative definition, give it a description, and define a MIME (Multi-Purpose Internet Mail Extension) type that is used to ensure that the browsers know what kind of component this is. For example, the company corporate logo could be an *image/gif* MIME type, indicating that it is a .gif file that should be rendered in the browser.

You can create, modify, and delete resources using the Business Modeler application. You can also use standard MQL commands such as create, delete, copy, modify, print, and list to manage resources. Specialized functions and embedded commands facilitate evaluation, translation, or formatting of objects or output on HTML pages.

Resource objects currently are not supported in a J2EE environment.

Defining a Resource Object

As a Business Administrator, you can create new resource objects that can be used in Web page design. A resource object requires a file and MIME type, which defines the content of the file.

To define a resource object

1. Click 🖼 or select **Resource** from the Object>New menu.

   The New Resource dialog box displays, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Defining a Name

You must specify a unique name for each resource that you create. You should assign a name that has meaning to both you and the user. The name you choose is the name that will be referenced to include this resource within a Web page. The resource name is limited to 127 characters.

Assigning an Icon

You can assign a special icon to the new resource. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

Type a description for the resource. The description provides general information for you and the user about the function, use, or content of the resource.

The description can consist of a prompt, comment, or qualifying phrase. There is no limit to the number of characters you can include in the description.

For example, if you were defining a resource named “Corporate Logo", you might write a Description similar to one of the following.

`Small size logo for use in page footer`
`Large size logo for use in banners`

Specifying the MIME type

You can associate a MIME (Multi-Purpose Internet Mail Extension) type with a resource, which defines the content type of the file.

The format of MIME types is a type and subtype separated by a slash, for example, image/gif or video/mpeg.

The major MIME types are application, audio, image, text, and video. There are a variety of formats that use the application type. For example, application/x-pdf refers to Adobe Acrobat Portable Document Format files. For information on specific MIME types (or more appropriately-called “media” types) refer the Internet Assigned Numbers Authority

Web site at http://www.isi.edu/in-notes/iana/assignments/media-types/. The IANA is the repository for assigned IP addresses, domain names, protocol numbers, and has also become the registry for a number of Web-related resources including media types.

Date and Time

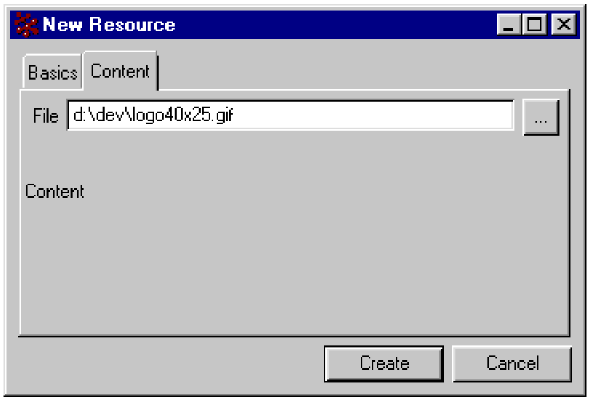
The system automatically includes the date and time that the resource object is originally created, and the date and time of the last modification.

Defining the Hidden Option

You can mark the new resource as "hidden" so that it does not appear in the chooser in Matrix Navigator. Users who are aware of the hidden resource's existence can enter its name manually where appropriate. Hidden objects are also accessible through MQL.

Defining Resource Content

Select the **Content** tab to display the resource content:



File d:\dev\logo40x25.gif

In the **File** text box, type the name of the binary file to be used as a resource or click the Browse button to choose a file from the browser. Resources are often .gif images, but they could be movie or video files or any resource that you use in a Web application, including:

- GIF
- JPEG
- MPEG
- AVI
- WAV
- JAR
- CAB

20

Working With Forms

Overview

A *form* is a window in which information related to an object is displayed. The Business Administrator designs the form, determining the information to be presented as well as the layout. Forms are created for specific object types, since the data contained in different types can vary greatly. In addition, one object type can have several forms associated with it, each displaying different collections of information. When a user attempts to display information about an object by using a form, Live Collaboration only offers those forms that are valid for the selected object.

Expressions can be used in the form’s field definitions to navigate the selected object’s connections and display information from the relationship (attributes) or from the objects found at their ends. Forms can be used as a means of inputting or editing the attributes of the selected object. However, attributes of related objects cannot be edited in a form.

Designing a Form

You design the layout and content of a form and then store the form as a Form definition.

To define a form

1. Click 🗒 or select **Form** from the Object>New menu.

   The New Form dialog box opens, showing the **Basics** tab.

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow. While only the **Name**, **Type**, **Units**, **Width**, and **Height** are required, the other parameter values can further define the form.

Defining a Name

Enter a unique name for the form. The form name is limited to 127 characters. For additional information, refer to *Administrative Object Names* in Chapter 1.

Assigning an Icon

You can assign a special icon to the new form. Icons help users locate and recognize items. When assigning an icon, you must select a GIF format file, as described in *Defining an Icon* in Chapter 1.

Defining a Description

A description provides general information about the function of the form.

There is no limit to the number of characters you can include in the description. However, keep in mind that the description appears when the mouse pointer stops over the form in a chooser. Although you are not required to provide a description, this information is helpful when a choice is requested.

Assigning a Type

Indicate the object types with which the form is associated.

To associate object types with the form

- Type the type name in the **Type** text box.

  Or

- Click the **Type** ellipsis button and select an object type from the Type Chooser that appears.

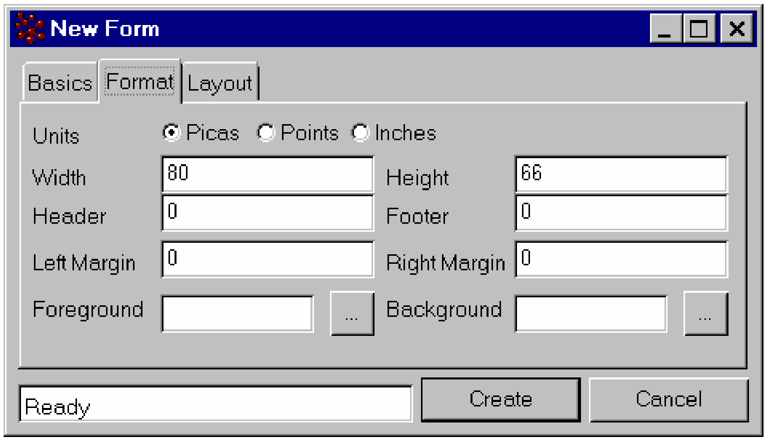
Defining the Hidden Option

You can mark the new form as “hidden” so that it does not appear in the Forms chooser in Live Collaboration. You may want to use the hidden option if, for example, an object is under development or if it is intended only for your personal use. Hidden objects are accessible through MQL.

Defining Format

To specify the format

1. Click **Format** tab from the New Form dialog box.



Units ● Picas ○ Points ○ Inches

Width 80 Height 66

Header 0 Footer 0

Left Margin 0 Right Margin 0

Foreground Background

Ready

2. Assign values to the parameters, as appropriate.

   Each parameter and its possible values are discussed in the sections that follow.

Indicating Units

Indicate the units of page measurement. Choose **Picas** (fixed size characters that render 80 by 66 on an 8-1/2 x 11 sheet), **Points** (graphical units of 72 points per inch), or **Inches**.

When you are viewing a form definition, as described in *Viewing an Administrative Object* in Chapter 1, you can select a different unit type (Picas, Points, or Inches) and the measurements change accordingly.

Defining Width and Height

Define the page dimensions of the form. A page can be any size that your printer can handle. Enter values for the **Width** and **Height** of the page. The values are measured in the units selected.

Defining a Header and Footer

Define the amount of space at the top (**Header**) and bottom (**Footer**) of each page. The values are measured in the units selected.

Defining Margins

Margins are the space at the left and right edges of the page. Enter values for the margins, as appropriate. The values are measured in the units selected.

Defining Color

Define the color of the **Foreground** (text) and the **Background** of the form. Type the name of a color in each of the text boxes or use the ellipsis buttons to select a color from a list of colors. Choose foreground and background colors that complement each other. Colors that offer high contrast together work best.

Creating Fields on a Form

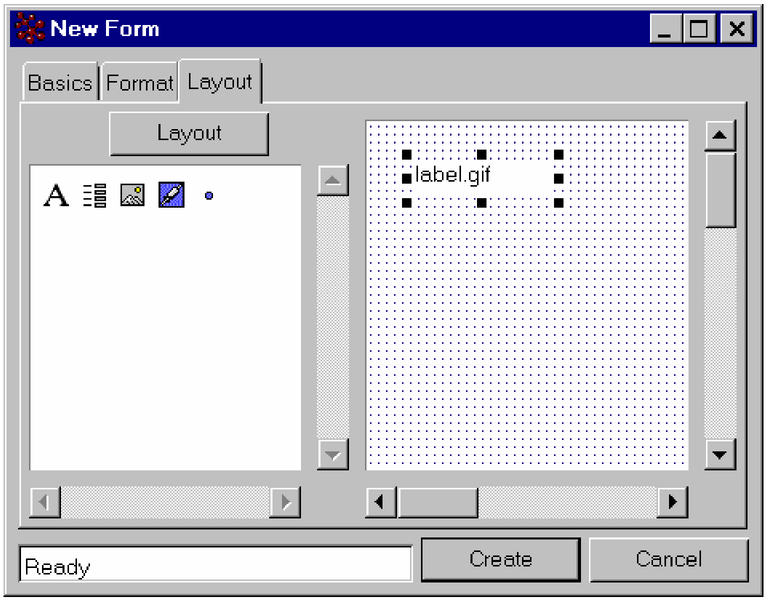
After entering the appropriate information in each text box, you are ready to create fields on the form.

To create the fields on a form

1. Click the **Layout** tab in the New Form dialog box.
2. Select a field type in the list box to the left of the form grid.

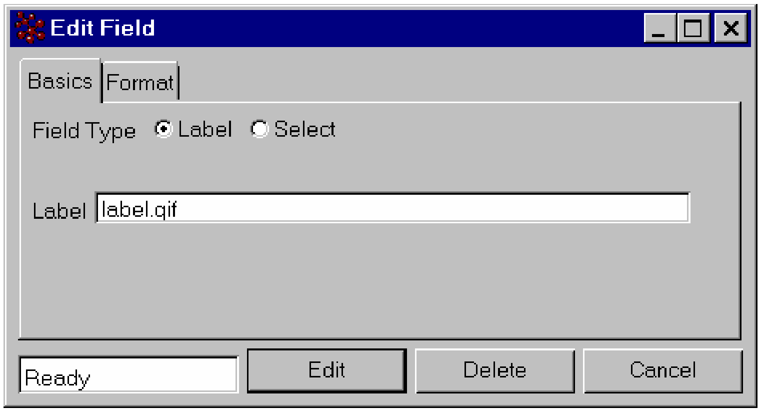
   The types are described in the table in step 5.
3. Drag the field type from the list box and drop it on the appropriate location on the form grid.

   The field type is displayed in a white area on the grid with the field type indicated if the type is text or as an icon if the type is graphical. For example:



4. Double-click a field area to edit the field.

   The Edit Field dialog box is displayed, showing the **Basics** tab. The contents of the Edit Field dialogs will vary depending on the type of field being edited.

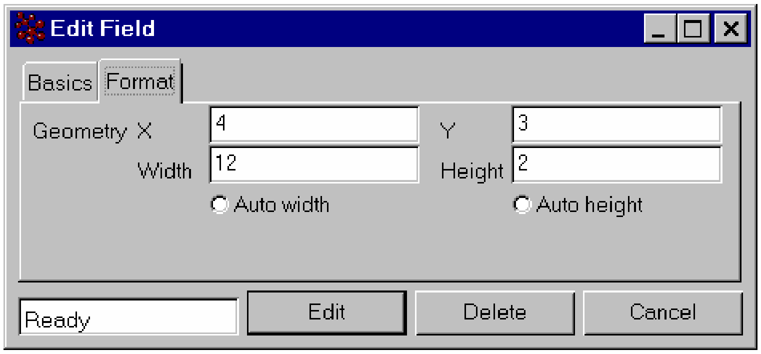


Field Type ● Label ○ Select

Label label.gif

Ready

Click the **Format** tab to see other options.



Geometry X 4 Y 3

Width 12 Height 2

○ Auto width ○ Auto height

Ready

5. Enter information about the field on these dialogs.

The following table describes each type of entry that you can make for the field.

Field	`Label`	A printable string of characters. Replace <label.gif> in the Label text box with text that should appear on the form.
	`Select`	Any selectable business object value. This allows for information to be retrieved from related objects as well as from the object to which the form is attached. Enter an expression that should appear on the form. For example, enter `NAME` to display the object name on the form.
	`Graphic`	An imported graphical image, such as a logo or scanned form. Enter the directory path for the graphic file. The graphic must be in GIF format.
Geometry	`X and Y`	The field location on the form. The X and Y coordinates identify the field's starting point—where the first character or the field value is displayed. The coordinates must be greater than 0.
	`Width and Height`	The field size. Enter a width value as the horizontal size of the field. Enter a height value as the vertical size.
`Foreground`		The color of the foreground (printed) information for the field. Enter the name of a color or click the ellipsis button to select from a list of colors.
`Background`		The background color of the field. Enter the name of a color or click the ellipsis button to select from a list of colors.

`Font`	The font in which the text of a Label or Select field appears. Enter the name of a font or click the ellipsis button to select from a list of fonts.
`Draw Border`	A drawing border placed around the field.
`Multiline`	When turned on, the returned value for the field displays on more than one line, as necessary. This option is available only if you are using the Select field type.
`Editable`	When turned on, the user can change the value while in the form. This option is available only if you are using the Select field type. *Only users with Modify Access can edit the fields.*

To move a field on the grid

1. Click on the middle of the white field area on the grid.
2. Drag the field to a new location.

To change the size of a field on the grid

1. Click on the desired edge of the white field area on the grid.
2. Drag to expand or reduce its size.

Deleting a Field

You can delete a field using the **Delete** button in the Edit Field dialog box.

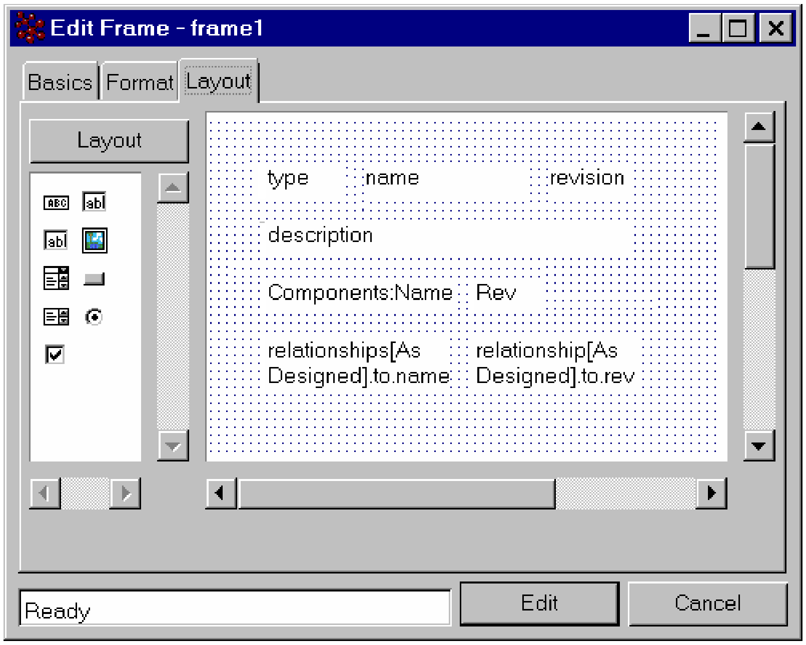
Creating the Form

After you have placed all appropriate field types and information on the grid, click **Create** in the New Form dialog box to create the form.

A Note About Select Statements

When using the Select field type, valid expressions include attributes, descriptions, and other selectable items that can be described in a manner similar to “where” clauses in an MQL query. For more information, see *Configuration Guide : Appendix: Selectables*.

For example, you might want to create a form for Assembly types called Components Required to list information about components related to a selected assembly. You might want to include the name, type, revision, and description of the selected Assembly object. You might also want to include the name, description, and revision of each Component object that is related to the selected Assembly object. The following figure shows how you would create this form.



When retrieving information from related objects, the number of values returned is the number of objects connected by the specified relationship (As Designed, in the example above) to the selected object. When creating a form, be sure the field size is large enough to handle multiple entries.

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z