PROGRESS®
ORBIX®

TS Thread Library Reference

Version 6.3.5, July 2011

PROGRESS software
BUSINESS MAKING PROGRESS™

Third Party Acknowledgements:

Progress Orbix v6.3.5 incorporates Jakarata-struts 1.0.2 from the Apache Software Foundation (http://www.apache.org). Such Apache Technology is subject to the following terms and conditions: The Apache Soft ware License, Version 1.1 Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved. Redistribution and use in source and

LOSS OF USE, DATA, OR PROFITS; OR BUSI NESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Founda tion. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates Jakarat-regexp 1.2 from the Apache Software Foundation (http://www.apache.org). Such Apache Technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 Copyright (c) 1999 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistri bution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "The Jakarta Project", "Jakarta -Regexp", and "Apache Software Foundation" and "Apache BCEL" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIA BLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates the Jakarta-log4j 1.2.6 from the Apache Software Foundation (http://www.apache.org). Such Apache Technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 Copyright (c) 1999 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following dis claimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in

the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "log4j" and "Apache Software Foundation" and "Apache BCEL" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written per mission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABIL ITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUD ING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foun dation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates Ant 1.5 from the Apache Software Foundation (http://www.apache.org). Such technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 Copyright (c) 2000-2002 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the fol lowing disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Ant" and "Apache Software Foundation" and "Apache BCEL" must not be used to endorse or promote products derived from this software without prior writ ten permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MER CHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contri butions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates Xalan-j 2.3.1 from the Apache Software Foundation (http://www.apache.org). Such Apache Technology is subject to the following terms and conditions: The Apache Soft ware License, Version 1.1. Copyright (c) 1999 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Xalan" and "Apache Software Foundation" and "Apache BCEL" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MER CHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contri butions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates the Xerces-c++ 2.4 from the Apache Software Foundation (http://www.apache.org). Such Apache Technology is subject to the following terms and conditions: The Apache Software License, Version 1.1. Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redis tributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Xerces" and "Apache Software Foundation" and "Apache BCEL" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE

APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Founda tion. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates xerces-j 2.5 from the Apache Software Foundation (http://www.apache.org). Such Apache Technology is subject to the following terms and conditions: The Apache Software License, Version 1.1. Copy right (c) 1999-2002 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTIC ULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates the Tomcat 4.0.4 from the Apache Software Foundation (http://www.apache.org). Such Apache Technology is subject to the following terms and conditions: The Apache Software License, Version 1.1. Copyright (c) 1999, 2000 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redis tributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the

redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "The Jakarta Project", "Tomcat" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foun dation, please see <http://www.apache.org/>.

Progress Orbix v6.3.5 incorporates MCPP 2.6.4 from the MCPP Project. Such technology is subject to the following terms and conditions: Copyright (c) 1998, 2002-2007 Kiyoshi Matsui kmatsui@t3.rim.or.jp All rights reserved. This software including the files in this directory is provided under the following license. Redistribu tion and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCURE MENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CON TRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Orbix v6.3.5 incorporates Xalan c++ v1.7 from The Apache Software Foundation. Such technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 Copyright (c) 1999-2004 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redis tributions in binary form must reproduce the above copyright notice, this list of conditions and the follow ing disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the follow-

ing acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Xalan" and "Apache Software Foundation" must not be used to endorse or promote prod ucts derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICU LAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==========

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

Progress Orbix v6.3.5 incorporates Tcl 8.4.15 from Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, and other parties. Such technology is subject to the following terms and conditions: This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files. The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply. IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THE AUTHORS AND DISTRIBUTORS SPE CIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WAR RANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFI CATIONS. GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the

foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

Progress Orbix v6.3.5 incorporates bzip2 1.0.2 from Julian Seward. Such Technology is subject to the following terms and conditions: This program, "bzip2" and associated library "libbzip2", are copyright (C) 1996-2002 Julian R Seward. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. 3. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software. 4. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. Julian Seward, Cambridge, UK.jseward@acm.org bzip2/libbzip2 version 1.0.2 of 30 December 2001.

Progress Orbix v6.3.5 incorporates zlib 1.2.3 from Jean-loup Gailly and Mark Adler. Such Technology is subject to the following terms and conditions: License /* zlib.h -- interface of the 'zlib' general purpose compression library version 1.2.3, July 18th, 2005 Copyright (C) 1995-2000 Jean-loup Gailly and Mark Adler. This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions: 1. The origin of this software must not be mis represented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software. 3. This notice may not be removed or altered from any source distribution. Jean-loup Gailly jloup@gzip.org Mark Adler madler@alumni.caltech.edu */

Progress Orbix v6.3.5 incorporates the MinML 1.7 from John Wilson. Such Technology is subject to the following terms and conditions: Copyright (c) 1999, John Wilson (tug@wilson.co.uk). All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice,, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following dis claimer in the documentation and/or other materials provided with the distribution. All advertising materials mention ing features or use of this software must display the following acknowledgement: This product includes software devel oped by John

Wilson. The name of John Wilson may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY JOHN WILSON "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JOHN WILSON BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABIL ITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Orbix v6.3.5 incorporates JDOM vbeta9 from JDOM. Such Technology is subject to the following terms and conditions: LICENSE.txt, v 1.10 2003/04/10 08:36:05 jhunter Exp $ Copyright (C) 2000-2003 Jason Hunter & Brett McLaughlin. All rights reserved. Redistribution and use in source and binary forms, with or with out modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer. 2. Redistribu tions in binary form must reproduce the above copyright notice, this list of conditions, and the dis claimer that follows these conditions in the documentation and/or other materials provided with the distribu tion. 3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <license AT jdom DOT org>. 4. Prod ucts derived from this soft ware may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management <pm AT jdom DOT org>. In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the soft ware itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)." Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WAR RANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DIS CLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIA BLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABIL ITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter AT jdom DOT org> and Brett McLaughlin <brett AT jdom DOT org>. For more information on the JDOM Project, please see <http://www.jdom.org/>.

Progress Orbix v6.3.5 incorporates OpenSSL 0.9.8i Copyright (c) 1998-2008 The OpenSSL Project Copyright (c) 1995-1998 Eric A. Young, Tim J. Hudson All rights reserved. Such Technology is subject to the following terms and conditions: The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to

OpenSSL please contact openssl-core@openssl.org. OpenSSL License - Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted pro vided that the following conditions are met: 1. Redistributions of source code must retain the above copy right notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)" 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org. 5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. 6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)" THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAM AGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERV ICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This product includes cryp tographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). - Original SSLeay License - Copyright (C) 1995-1998 Eric Young (eay@crypt soft.com) All rights reserved. This package is an SSL implementation written by Eric Young (eay@crypt soft.com). The implementation was written so as to conform with Netscapes SSL. This library is free for commercial and non-commer cial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com). Copy right remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package. Redistri bution and use in source and binary forms, with or with out modification, are permitted provided that the follow ing conditions are met: 1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of con ditions and the following dis claimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mention ing features or use of this software must display the following acknowledge ment: "This product includes crypto graphic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the rou tines from the library being used are not crypto graphic related :-). 4. If you include any Windows specific code (or a deriv ative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)" THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPE CIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCURE MENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSI BILITY OF SUCH DAMAGE. The licence and distribution terms for any publically available version or deriva tive of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

Progress Orbix v6.3.5 incorporates PCRE v7.8 from the PCRE Project. Such Technology is subject to the following terms and conditions:

PCRE LICENCE

PCRE is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language. Release 7 of PCRE is distributed under the terms of the "BSD"licence, as specified below. The documentation for PCRE, supplied in the "doc" directory, is distributed under the same terms as the software itself. The basic library functions are written in C and are freestanding. Also included in the distribution is a set of C++ wrapper functions.

THE BASIC LIBRARY FUNCTIONS

Written by: Philip Hazel

Email local part: ph10

Email domain: cam.ac.uk

University of Cambridge Computing Service,

Cambridge, England.

Copyright (c) 1997-2008 University of Cambridge

All rights reserved.

THE C++ WRAPPER FUNCTIONS

Contributed by: Google Inc.

Copyright (c) 2007-2008, Google Inc.

All rights reserved.

THE "BSD" LICENCE

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the University of Cambridge nor the name of

list.c) covered by a similar license but with different Copyright notices: Copyright (C) 1998-2002 Daniel Veillard. All Rights Reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including with out limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Soft ware, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTA BILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIA BILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

=== trio.c, trio.h: Copyright (C) 1998 Bjorn Reese and Daniel Stenberg. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THIS SOFTWARE IS PROVIDED "AS IS" AND WITH OUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE AUTHORS AND CONTRIB UTORS ACCEPT NO RESPONSIBILITY IN ANY CONCEIVABLE MANNER. ==== triop.h: Copyright (C) 2000 Bjorn Reese and Daniel Stenberg. Permission to use, copy, modify, and dis tribute this software for any purpose with or without

fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTIC ULAR PURPOSE. THE AUTHORS AND CONTRIBUTORS ACCEPT NO RESPONSIBILITY IN ANY CONCEIVABLE MANNER.

==== hash.c: Copyright (C) 2000 Bjorn Reese and Daniel Veillard. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permis sion notice appear in all copies. THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHAN TIBILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE AUTHORS AND CONTRIBUTORS ACCEPT NO RESPONSIBILITY IN ANY CONCEIVABLE MANNER.

===== list.c: Copyright (C) 2000 Gary Pennington and Daniel Veillard. Permission

to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE AUTHORS AND CONTRIBUTORS ACCEPT NO RESPONSIBILITY IN ANY CONCEIVABLE MANNER. ===

triodef.h, trionan.c, trionan.h: Copyright (C) 2001 Bjorn Reese Permission to use, copy, modify, and distribute this soft ware for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permis sion notice appear in all copies. THIS SOFTWARE IS PROVIDED ``AS IS'' AND

Updated: 14-Jul-2011

Contents

Table of Contents

Threading and Synchronization Toolkit Overview

The Threading and Synchronization (TS) toolkit provides an object-oriented and platform-neutral abstraction that hides the diverse, lower-level, thread packages. Table 1 shows the threading and synchronization (TS) classes organized into some useful groups.

Table 1: *TS Thread Classes*

Thread Management	IT_CurrentThread IT_Thread IT_ThreadBody IT_ThreadFactory IT_TerminationHandler IT_TSVoidStar
Thread Errors and Exceptions	IT_TSBadAlloc IT_DefaultTSErrorHandler IT_TSError IT_TSErrorHandler IT_TSLogic IT_TSRuntime
Mutex Locks	IT_Locker IT_Mutex IT_PODMutex IT_RecursiveMutex IT_RecursiveMutexLocker
Thread Synchronization	IT_Condition IT_Gateway IT_Semaphore IT_TimedCountByNSemaphore IT_TimedOneshot IT_TimedSemaphore

The rest of this overview covers these topics:

Timeouts

Timeouts are expressed in milliseconds. They represent the time period from the invocation of the timed method until the expiration of the timer. This time-out period is approximate because it is affected by the number and kind of interrupts received and by the changes external sources may make to the system’s time.

Execution Modes

The TS classes are designed to be efficient and to help you write code that is correct and portable across various platforms. You can build TS applications in either of the following modes:

Unchecked	This is the normal production mode. Inexpensive checks, such as checking values returned by the API, are performed, but a minimum of memory, locking, and system calls are used to implement TS features.
Checked	In this mode, extra-checking is performed to detect erroneous or non-portable situations. On platforms that support exceptions, exceptions are raised to report such errors. This mode may be less time or space efficient than the unchecked mode.

The effect of a program that runs correctly (the program does not create any TS error object) in the checked mode is identical to that of the unchecked mode.

TS provides two kinds of classes in different sets of header files. These include wrapper and inline classes.

Wrapper Classes

Wrapper classes are the recommended classes to use because you can switch between checked and unchecked modes by simply re-linking without recompiling your application. These clean, platform-neutral wrapper classes simply delegate to the appropriate inlined classes for whichever mode you are using.

The wrapper classes are in header files ending in `.h`.

Inlined Classes

To minimize the delegation overhead of wrapper classes, the TS toolkit also provides C++ classes with only inlined member methods and pre-preprocessor directives. These inline classes accommodate the differences between the underlying thread packages.

Delegation overhead for a normal method call is generally negligible, but you can save on this overhead by using these inlined classes directly. However by using these header files, you will need to recompile your application whenever you want to switch between checked and unchecked modes, and each time even minor improvements are made to the TS implementation.

The inline classes are in header files ending in `_i.h`.

Setting an Execution Mode

Table 2 shows the default settings for each platform.

Table 2: *Default Thread Settings*

Platform	Thread Primitives	Default Mode
HPUX 11 Solaris 2.6	Posix	unchecked
HPUX 10.20	DCE	unchecked
Other Solaris	UI	unchecked
Win32	Win32	unchecked

To set a different mode, you reset the library by inserting the preferred `lib` subdirectory at the beginning of your `LD_LIBRARY_PATH` or `SHLIB_PATH`. For example, to reset to the checked mode, do the following for your respective platform:

Solaris	Put the following at the beginning of your `LD_LIBRARY_PATH`: `/vob/common/ts/lib/posix/checked`
HPUX 10.20	Put the following at the beginning of your `SHLIB_PATH`: `/vob/common/ts/lib/dce/checked`
HPUX 11.00	Put the following at the beginning of your `SHLIB_PATH`: `/vob/common/ts/lib/posix/checked`
NT	Put the following at the beginning of your `PATH`: `/common/ts/lib/win32/checked`

Errors and Exceptions

Table 3 summarizes the TS error classes:

Table 3: *Error and Exception Classes*

Control	Exceptions
`IT_DefaultTSErrorHandler`	`IT_TSBadAlloc`
`IT_TSError`	`IT_TSLogic`
`IT_TSErrorHandler`	`IT_TSRuntime`

The TS API allows you to use either error parameters or exceptions. The last parameter of almost every TS method is a reference to an error handler object of the class `IT_TSErrorHandler`. When a TS method detects an error, it creates an `IT_TSError` object and passes it to `IT_TSErrorHandler::handle()`.

TS errors form the hierarchy shown in Figure 1. An `IT_TSRuntime` error generally signals an error detected by the operating system or the underlying thread package. An `IT_TSLogic` error reports a logic error in your program, for example, when a thread tries to release a lock it does not own. Logic errors are either detected by the underlying thread package, or by extra checking code in checked mode. An `IT_TSBadAlloc` error signals that the `new` operator failed.

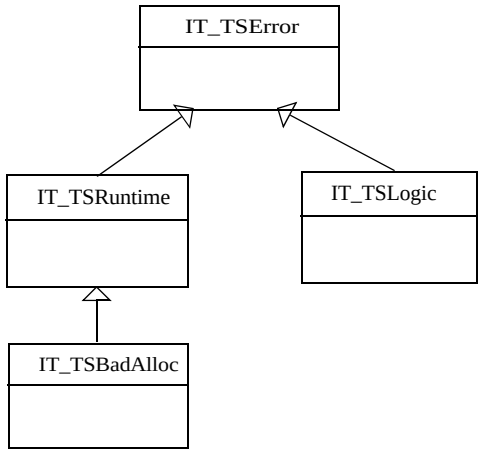

Figure 1: *The TS Error Class Hierarchy*

The TS API provides a default, static, and stateless error handler named `IT_DefaultTSErrorHandler`. If you use exceptions, this error handler throws `IT_TSError` objects. In environments that do not use exceptions this handler aborts the process.

For most applications, the default error handler object provides the desired behavior. In this situation, instead of passing an `IT_DefaultTSErrorHandler` object each time you call a TS method, you can define in your build command the environment variable `IT_TS_DEFAULTED`. This will instruct the TS API to use the default error handler object for the error handler parameter. For example:

```
#ifndef IT_TS_DEFAULT_ERROR_HANDLER
#ifdef IT_TS_DEFAULTED
#define IT_TS_DEFAULT_ERROR_HANDLER = IT_DefaultTSErrorHandler
#else
#define IT_TS_DEFAULT_ERROR_HANDLER
#endif
#endif
```

C++ destructors do not have parameters, and as result, cannot be given an error handler object parameter. In the checked mode, the TS API reports errors in destructors to the default error handler object. In the unchecked mode, the TS API does not report errors that occur in destructors.

Because default parameters are not part of the function-type in C++, the TS library can be built with or without defining `IT_TS_DEFAULTED`. Also, the same library can be used by modules that use the defaulted parameter and by modules built without defining `IT_TS_DEFAULTED`.

If you intend to use your own error handler objects in your application, it is strongly recommended that you do not define `IT_TS_DEFAULTED` to avoid using the default error handler object by mistake. If you want to consistently use the same error handler object, you can define `IT_TS_DEFAULT_ERROR_HANDLER` in your command or in a non-exported file. For example:

```
#define IT_TS_DEFAULT_ERROR_HANDLER = myErrorHandler;
```

IT_Condition Class

The `IT_Condition` class provides a signalling mechanism that events use to synchronize when sharing a mutex. In one atomic operation, a condition wait both releases the mutex and waits until another thread signals or broadcasts a change of state for the condition.

```
class IT_Condition {
public:
    IT_Condition(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    ~IT_Condition();
    void wait(
        IT_Mutex& app_mutex,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    void wait(
        IT_MutexLocker& locker,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    void signal(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    void broadcast(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
};
```

IT_Condition::broadcast()

```
void broadcast(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Wakes up all waiting threads. One thread acquires the mutex and resumes with the associated mutex lock. The rest of the threads continue waiting.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

See Also `IT_Mutex`

IT_Condition::IT_Condition() Constructor

```
IT_Condition(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

The constructor for an `IT_Condition` object.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_Condition::~IT_Condition() Destructor

```
~IT_Condition();
```

The destructor for an `IT_Condition` object.

Enhancement Orbix enhancement.

IT_Condition::signal()

```
void signal(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Wakes up a single waiting thread. The thread resumes with the associated mutex locked.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_Condition::wait()

```
void wait(
    IT_Mutex& app_mutex,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);

void wait(
    IT_MutexLocker& locker,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Atomically releases the mutex, and waits until another thread calls `signal()` or `broadcast()`.

Parameters

`app_mutex`	Use the mutex `app_mutex`.
`locker`	Use the mutex in `locker`.
`eh`	

The mutex must always be locked when `wait()` is called. When a condition wakes up from a wait, it resumes with the mutex locked.

Enhancement Orbix enhancement.

IT_CurrentThread Class

The `IT_CurrentThread` class gives access to the current thread. It has only static member methods.

```
class IT_TS_API IT_CurrentThread {
public:
    static IT_Thread self(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    static int is_main_thread();

    static void cleanup();

    static void yield();

    static void sleep(
        unsigned long milliseconds
    );

    static long id();
};
```

IT_CurrentThread::cleanup()

```
static void cleanup();
```

Cleans up thread-specific data. A thread typically calls `cleanup()` before exiting. Threads created with an `IT_ThreadFactory` do this automatically.

Enhancement Orbix enhancement.

IT_CurrentThread::id()

```
static long id();
```

Returns a unique identifier for the current thread.

Enhancement Orbix enhancement.

IT_CurrentThread::is_main_thread()

```
static int is_main_thread();
```

Returns 1 if the caller is the main thread, but returns 0 if it is not.

Enhancement Orbix enhancement.

IT_CurrentThread::self()

```
static IT_Thread self(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Returns an IT_Thread object for the thread that calls this method.

Parameters

eh	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_CurrentThread::sleep()

```
static void sleep(
    unsigned long milliseconds
);
```

Suspends the current thread for the approximate number of milliseconds input.

Parameters

milliseconds	The length of time in milliseconds to suspend the thread.

Enhancement Orbix enhancement.

IT_CurrentThread::yield()

```
static void yield();
```

Yields the CPU to another thread of equal priority, if one is available.

Enhancement Orbix enhancement.

IT_DefaultTSErrorHandler Class

The `IT_DefaultTSErrorHandler` class is the default TS error handler. If you use exceptions, this error handler throws `IT_TSError` objects. In environments that do not use exceptions this handler aborts the process.

```
class IT_DefaultTSErrorHandler : public IT_TSErrorHandler{
public:
    virtual ~IT_DefaultTSErrorHandler()
    virtual void handle(
        const IT_TSError& this_error
    );
};
```

See page 4 for more on error handling.

IT_DefaultTSErrorHandler::handle()

```
void handle(
    const IT_TSError& this_error
);
```

Do appropriate processing for the given error.

Parameters

`this_error` A reference to an error object.

Enhancement Orbix enhancement.

IT_DefaultTSErrorHandler::~IT_DefaultTSErrorHandler() Destructor

```
~IT_DefaultTSErrorHandler()
```

The destructor for the error handler object.

Enhancement Orbix enhancement.

IT_Gateway Class

The `IT_Gateway` class provides a gate where a set of threads can only do work if the gate is open.

```
class IT_Gateway {
public:
    IT_Gateway(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    ~IT_Gateway();

    void open(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void close(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void wait(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

private:
...
```

IT_Gateway::close()

```
void close(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Close the gateway so no threads can do any work.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_Gateway::IT_Gateway() Constructor

```
IT_Gateway(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

The gateway constructor.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_Gateway::~IT_Gateway() Destructor

```
~IT_Gateway();
```

The destructor.

Enhancement Orbix enhancement.

IT_Gateway::open()

```
void open(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Open the gateway to allow threads to work.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_Gateway::wait()

```
void wait(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Wait for a thread to finish.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_Locker Template Class

`IT_Locker` is a helper class for locking and unlocking non-recursive mutexes, including `IT_Mutex` and `IT_PODMutex` objects. Typically a locker locks a mutex in its constructor and releases it in its destructor. This is particularly useful for writing clean code that behaves properly when an exception is raised.

An `IT_Locker` object must be created on the stack of a particular thread, and must never be shared by more than one thread.

The `IT_Locker` method definitions are inlined directly in the class declaration, because these methods call each other. If a definition calls a method that is not previously declared inlined, this method is generated out of line, regardless of its definition (which can be provided later in the translation unit with the inline keyword).

```
template<class T> class IT_Locker {
public:
    IT_Locker(
        T& mutex,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    ) :
        m_mutex(mutex),
        m_locked(0),
        m_error_handler(eh)
        {
            lock();
        }

    IT_Locker(
        T& mutex,
        int wait,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    ) :
        m_mutex(mutex),
        m_locked(0),
        m_error_handler(eh)
        {
            if (wait)
            {
```

```
                lock();
            }
            else
            {
                trylock();
            }
        }

    ~IT_Locker()
    {
        cancel();
    }

    void cancel()
    {
        if (m_locked)
        {
            m_mutex.unlock(m_error_handler);
            m_locked = 0;
        }
    }

    int is_locked()
    {
        return m_locked;
    }

    void lock()
    {
        m_mutex.lock(m_error_handler);
        m_locked = 1;
    }

    int trylock()
    {
        return (m_locked = m_mutex.trylock(m_error_handler));
    }

    T& mutex()
    {
        return m_mutex;
    }
private:
```

```
...
```

IT_Locker::cancel()

```
void cancel() {
    if (m_locked)
    {
        m_mutex.unlock(m_error_handler);
        m_locked = 0;
    }
}
```

Releases the mutex only if it is locked by this locker. You can call `cancel()` safely even when the mutex is not locked.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSRuntime
IT_TSLogic

IT_Locker::is_locked()

```
int is_locked() {
    return m_locked;
}
```

returns 1 if this mutex locker has the lock and returns 0 if it does not.

Enhancement Orbix enhancement.

IT_Locker::IT_Locker()

```
IT_Locker(
    T& mutex,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
) :
    m_mutex(mutex),
    m_locked(0),
    m_error_handler(eh)
```

```
{
    lock();
}
```

A constructor for a locker object that locks the given mutex.

```
IT_Locker(
    T& mutex,
    int wait,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
) :
    m_mutex(mutex),
    m_locked(0),
    m_error_handler(eh)
{
    if (wait)
    {
        lock();
    }
    else
    {
        trylock();
    }
}
```

A constructor for a locker object.

Parameters

`mutex`	The mutex to which the locker applies.
`wait`	If wait has a value of 1, this constructor waits to acquire the lock. If wait has a value of 0, the constructor only tries to lock the mutex.
`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

See Also `IT_Locker::trylock()`

IT_Locker::~IT_Locker()

```
~IT_Locker()
{
```

```
    cancel();
}
```

The destructor releases the mutex if it is locked by this locker.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSLogic
IT_TSRuntime

IT_Locker::lock()

```
void lock()
{
    m_mutex.lock(m_error_handler);
    m_locked = 1;
}
```

Locks the mutex associated with the locker.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSLogic
IT_TSRuntime

IT_Locker::mutex()

```
T& mutex()
{
    return m_mutex;
}
```

Returns direct access to the locker's mutex.

Enhancement Orbix enhancement.

IT_Locker::trylock()

```
int trylock()
{
    return (m_locked = m_mutex.trylock(m_error_handler));
}
```

Tries to lock the mutex. Returns 1 if the mutex is successfully locked or 0 if it is not locked.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSLogic
IT_TSRuntime

IT_Mutex Class

An IT_Mutex object is a synchronization primitive for mutual exclusion locks.

When a thread has successfully locked, it is said to own the IT_Mutex. IT_Mutex objects have scope only within a single process (they are not shared by several processes) and they are not recursive. When a thread that owns an IT_Mutex attempts to lock it again, a deadlock occurs.

You use an IT_Mutex in conjunction with an IT_Locker object to lock and unlock your mutexes.

```
class IT_Mutex {
public:
    IT_Mutex(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    ~IT_Mutex();

    void lock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void unlock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    int trylock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

private:
// …
};
```

See Also

IT_Locker
IT_RecursiveMutex

IT_Mutex::IT_Mutex() Constructor

```
IT_Mutex(I
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Constructs an IT_Mutex object. It is initially unlocked.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions The `IT_TSRuntime` error can be reported.

IT_Mutex::~IT_Mutex() Destructor

```
IT_Mutex();
```

The destructor for the mutex.

Enhancement Orbix enhancement.

IT_Mutex::lock()

```
void lock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Blocks until the `IT_Mutex` can be acquired.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

`IT_TSLogic`
`IT_TSRuntime`

IT_Mutex::trylock()

```
int trylock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Tries to acquire the lock. If successful, the method returns a 1 immediately, otherwise it returns a 0 and does not block.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSLogic
IT_TSRuntime

IT_Mutex::unlock()

```
void unlock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Releases this `IT_Mutex`. Only the owner thread of an `IT_Mutex` is allowed to release an `IT_Mutex`.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSLogic
IT_TSRuntime

IT_PODMutex Structure

An IT_PODMutex is a mutex for a “plain old data” (POD) structure. Just as with a standard C++ PODS, an IT_PODMutex can be fully initialized at compile time without the overhead of an explicit constructor call. This is particularly useful for static objects. Likewise, the object can be destroyed without an explicit destructor call (in a manner similar to the C language).

You can use the built-in definition IT_POD_MUTEX_INIT to easily initialize an IT_PODMutex to zero. For example:

```
static IT_PODMutex my_global_mutex = IT_POD_MUTEX_INIT;
```

You use an IT_PODMutex in conjunction with an IT_Locker object to lock and unlock your mutexes. The structure members for an IT_PODMutex include the following:

```
struct IT_TS_API IT_PODMutex {
    void lock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    int trylock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    void unlock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    // DO NOT USE and DO NOT MAKE PRIVATE
    unsigned char m_index;
};
```

See Also

IT_Locker
IT_Mutex

IT_PODMutex::lock()

```
void lock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Blocks until the mutex can be acquired.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

`IT_TSLogic`
`IT_TSRuntime`

IT_PODMutex::m_index Data Type

```
unsigned char m_index;
```

Note: For internal use only.

IT_PODMutex::trylock()

```
int trylock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Tries to acquire the mutex lock. If `trylock()` succeeds, it returns a 1 immediately. Otherwise it returns 0.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

`IT_TSLogic`
`IT_TSRuntime`

IT_PODMutex::unlock()

```
void unlock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Releases the mutex lock. Only the owner of a mutex is allowed to release it.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSLogic
IT_TSRuntime

IT_RecursiveMutex Class

An `IT_RecursiveMutex` object is a synchronization primitive for mutual exclusion. In general do not used it directly.

Note: It is strongly recommended that you use the `IT_RecursiveMutexLocker` to lock and unlock your recursive mutexes.

In most respects an `IT_RecursiveMutex` object is similar to an `IT_Mutex` object. However, it can be locked recursively, which means that a thread that already owns a recursive mutex object can lock it again in a deeper scope without creating a deadlock condition.

When a thread has successfully locked a recursive mutex, it is said to own it. Recursive mutex objects have process-scope which means that they are not shared by several processes.

To release an `IT_RecursiveMutex`, its owner thread must call `unlock()` the same number of times that it called `lock()`.

```
class IT_RecursiveMutex {
public:
    IT_RecursiveMutex(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    ~IT_RecursiveMutex();

    void lock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void unlock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    int trylock(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
```

```
    );
private:
...
```

See Also IT_Mutex
IT_RecursiveMutexLocker

IT_RecursiveMutex::IT_RecursiveMutex() Constructor

```
IT_RecursiveMutex(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Constructs an `IT_RecursiveMutex` object. It is initially unlocked.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions The `IT_TSRuntime` error can be reported.

IT_RecursiveMutex::~IT_RecursiveMutex() Destructor

```
~IT_RecursiveMutex();
```

Destructor for an `IT_RecursiveMutex` object.

Enhancement Orbix enhancement.

IT_RecursiveMutex::lock()

```
void lock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Blocks until the recursive mutex can be acquired.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions The IT_TSRuntime error can be reported.

IT_RecursiveMutex::trylock()

```
int trylock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Tries to acquire the recursive mutex. If it succeeds, returns 1 immediately; otherwise returns 0.

Parameters

eh	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions The IT_TSRuntime error can be reported.

IT_RecursiveMutex::unlock()

```
void unlock(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Releases this recursive mutex (one count). Only the owner of a mutex is allowed to release it.

Parameters

eh	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSRuntime
IT_TSLogic

IT_RecursiveMutexLocker Class

The `IT_RecursiveMutexLocker` is a locker for recursive mutexes. The `IT_RecursiveMutexLocker` methods are defined as inline in the class declaration, because these methods call each other.

```
class IT_RecursiveMutexLocker {
public:
    IT_RecursiveMutexLocker(
        IT_RecursiveMutex& m,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    ) :
        m_recursive_mutex(m),
        m_lock_count(0),
        m_error_handler(eh)
    {
        lock();
    }

    IT_RecursiveMutexLocker(
        IT_RecursiveMutex& m,
        int wait,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    ) :
        m_recursive_mutex(m),
        m_lock_count(0),
        m_error_handler(eh)
    {
        if (wait)
        {
            lock();
        }
        else
        {
            trylock();
        }
    }

    ~IT_RecursiveMutexLocker()
```

```
{
    cancel();
}

void cancel()
{
    while (m_lock_count > 0)
    {
        m_recursive_mutex.unlock(m_error_handler);
        m_lock_count--;
    }
}

void lock()
{
    m_recursive_mutex.lock(m_error_handler);
    m_lock_count++;
}

unsigned int lock_count()
{
    return m_lock_count;
}

int trylock()
{
    if (m_recursive_mutex.trylock(m_error_handler) == 1)
    {
        m_lock_count++;
        return 1;
    }
    else
    {
        return 0;
    }
}

void unlock()
{
    m_recursive_mutex.unlock(m_error_handler);
    m_lock_count--;
}
```

```
    IT_RecursiveMutex& mutex()
    {
        return m_recursive_mutex;
    }

Private:
...
```

IT_RecursiveMutexLocker::cancel()

```
void cancel() {
    while (m_lock_count > 0)
    {
        m_recursive_mutex.unlock(m_error_handler);
        m_lock_count--;
    }
}
```

Releases all locks held by this recursive mutex locker. The `cancel()` method can be called safely even when the recursive mutex is not locked.

Enhancement Orbix enhancement.

IT_RecursiveMutexLocker::IT_RecursiveMutexLocker() Constructors

```
IT_RecursiveMutexLocker(
    IT_RecursiveMutex& m,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
) :
    m_recursive_mutex(m),
    m_lock_count(0),
    m_error_handler(eh)
{
    lock();
}
```

Constructs a recursive mutex locker object. This constructor locks the given recursive mutex.

```
IT_RecursiveMutexLocker(
    IT_RecursiveMutex& m,
```

```
    int wait,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
) :
    m_recursive_mutex(m),
    m_lock_count(0),
    m_error_handler(eh)
{
    if (wait)
    {
        lock();
    }
    else
    {
        trylock();
    }
}
```

Constructs a recursive mutex locker object.

Parameters

`m`	The mutex to which the locker applies.
`wait`	If wait has a value of 1, this constructor waits to acquire the lock. If wait has a value of 0, it only tries to lock the recursive mutex.
`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_RecursiveMutexLocker::~IT_RecursiveMutexLocker() Destructor

```
~IT_RecursiveMutexLocker()
{
    cancel();
}
```

The destructor releases all locks held by this recursive mutex locker.

Enhancement Orbix enhancement.

IT_RecursiveMutexLocker::lock()

```
void lock()
{
    m_recursive_mutex.lock(m_error_handler);
    m_lock_count++;
}
```

Acquires the lock.

Enhancement Orbix enhancement.

IT_RecursiveMutexLocker::lock_count()

```
unsigned int lock_count()
{
    return m_lock_count;
}
```

Returns the number of locks held by this recursive mutex locker.

Enhancement Orbix enhancement.

IT_RecursiveMutexLocker::mutex()

```
IT_RecursiveMutex& mutex()
{
    return m_recursive_mutex;
}
```

Returns direct access to the locker's recursive mutex.

Enhancement Orbix enhancement.

IT_RecursiveMutexLocker::trylock()

```
int trylock()
{
    if (m_recursive_mutex.trylock(m_error_handler) == 1)
    {
        m_lock_count++;
```

```
        return 1;
    }
    else
    {
        return 0;
    }
}
```

Tries to acquire one lock for the recursive mutex. Returns 1 if the mutex lock is successfully acquired or 0 if it is not.

Enhancement Orbix enhancement.

IT_RecursiveMutexLocker::unlock()

```
void unlock()
{
    m_recursive_mutex.unlock(m_error_handler);
    m_lock_count--;
}
```

Releases one lock held by this recursive mutex.

Enhancement Orbix enhancement.

IT_Semaphore Class

A semaphore is a non-negative counter, typically used to coordinate access to some resources.

```
class IT_Semaphore {
public:
    IT_Semaphore(
        size_t initialCount,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    ~IT_Semaphore();

    void post(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void wait(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    int trywait(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

private:
// …
};
```

IT_Semaphore::IT_Semaphore() Constructor

```
IT_Semaphore(
    size_t initialCount,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

A semaphore constructor that initializes the semaphore's counter with the value `initialCount`.

Parameters

`initialCount`	A positive integer value.
`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions The `IT_TSRuntime` error can be reported.

IT_Semaphore::~IT_Semaphore() Destructor

```
~IT_Semaphore();
```

Destroys the semaphore.

Enhancement Orbix enhancement.

IT_Semaphore::post()

```
void post(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Posts a resource thread with the semaphore. This method increments the semaphore's counter and wakes up a thread that might be blocked on `wait()`.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions The `IT_TSRuntime` error can be reported.

IT_Semaphore::trywait()

```
int trywait(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Tries to get a resource thread. The method returns 1 if it succeeds, and 0 if it fails.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is `IT_TSRuntime`.

IT_Semaphore::wait()

```
void wait(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Waits for one resource. The `wait()` method blocks if the semaphore's counter value is 0 and decrements the counter if the counter's value is greater than 0.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

`IT_TSLogic`
`IT_TSRuntime`

See Also `IT_TimedSemaphore`
`IT_TimedCountByNSemaphore`

IT_TerminationHandler Class

The `IT_TerminationHandler` class enables server applications to handle delivery of CTRL_C and similar events in a portable manner. On UNIX, the termination handler handles the following signals:

```
SIGINT
SIGTERM
SIGQUIT
```

On Windows, the termination handler is a wrapper around `SetConsoleCtrlHandler`, which handles delivery of the following control events:

```
CTRL_C_EVENT
CTRL_BREAK_EVENT
CTRL_SHUTDOWN_EVENT
CTRL_LOGOFF_EVENT
CTRL_CLOSE_EVENT
```

You can create only one termination handler object in a program.

```
#include <it_ts/ts_error.h>

typedef void (*IT_TerminationHandlerFunctionPtr)(long);

class IT_IFC_API IT_TerminationHandler
{
  public:

    IT_TerminationHandler(
       IT_TerminationHandlerFunctionPtr f,
       IT_ExceptionHandler& eh = IT_EXCEPTION_HANDLER
    );

    ~IT_TerminationHandler();
};
```

IT_TerminationHandler()

```
IT_TerminationHandler(
   IT_TerminationHandlerFunctionPtr f,
   IT_ExceptionHandler& eh = IT_EXCEPTION_HANDLER
);
```

Creates a termination handler object on the stack. On POSIX platforms, it is critical to create this object in the main thread before creation of any other thread, and especially before ORB initialization.

Parameters

`f` The callback function registered by the application. The callback function takes a single `long` argument:

- On UNIX, the signal number on Unix/POSIX
- On Windows, the type of event caught

~IT_TerminationHandler()

```
~IT_TerminationHandler();
```

Deregisters the callback, in order to avoid calling it during static destruction.

IT_Thread Class

An IT_Thread object represents a thread of control. An IT_Thread object can be associated with a running thread, associated with a thread that has already terminated, or it can be null, which means it is not associated with any thread.

The important class members are as follows:

```
class IT_Thread {
public:
    IT_Thread();

    ~IT_Thread();

    IT_Thread(
        const IT_Thread& other
    );

    IT_Thread& operator=(
        const IT_Thread& other
    );

    int operator==(
        const IT_Thread& x
    ) const;

    int operator!=(
        const IT_Thread& x
    ) const
        {
            return ! operator==(x);
        }

    int is_null() const;

    static void* const thread_failed;

    void* join(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    ) const;
```

```
    long id() const;
...
};
```

IT_Thread::id()

```
long id() const;
```

Returns a unique thread identifier. This method is useful for debugging.

Enhancement Orbix enhancement.

IT_Thread::is_null()

```
int is_null() const;
```

Tests if this is a null `IT_Thread` object.

Enhancement Orbix enhancement.

IT_Thread::IT_Thread() Constructors

```
IT_Thread(
    IT_Thread_i* t=0
);
```

Constructs a null `IT_Thread` object.

```
IT_Thread (
    const IT_Thread& other
);
```

Copies the `IT_Thread` object. This constructor does not start a new thread.

Parameters

`other`	The original thread to copy.

Enhancement Orbix enhancement.

IT_Thread::~IT_Thread() Destructor

```
~IT_Thread();
```

Destructor for an `IT_Thread` object.

Enhancement Orbix enhancement.

IT_Thread::join()

```
void* join(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
) const;
```

Waits until the thread has terminated and returns its exit status. At most one thread can successfully join a given thread, and only `Attached` threads can be joined. Note that even in the checked mode, `join()` does not always detect that you tried to join a `Detached` thread, or that you joined the same thread several times.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

`IT_TSLogic`
`IT_TSRuntime`

See Also `IT_CurrentThread`
`IT_ThreadBody`

IT_Thread::operator=()

```
IT_Thread& operator=(
    const IT_Thread& other
);
```

Assignment operator that copies the `IT_Thread` object. This does not start a new thread.

Parameters

`other`	The original thread that is copied.

Enhancement Orbix enhancement.

IT_Thread::operator==()

```
int operator==(
    const IT_Thread& x
) const;
```

Operator that checks if two `IT_Thread` objects refer to the same thread. Returns 1 if the two objects refer to the same thread or it returns 0 if they do not refer to the same thread.

Parameters

`x`	The thread to compare to this thread.

Enhancement Orbix enhancement.

IT_Thread::operator!=()

```
int operator!=(
    const IT_Thread& x
) const
```

Operator that checks if two `IT_Thread` objects refer to different threads. Returns 1 if the two objects refer to different threads or it returns 0 if they refer to the same thread.

Parameters

`x`	The thread to compare to this thread.

Enhancement Orbix enhancement.

IT_Thread::thread_failed Constant

```
static void* const thread_failed;
```

The constant `thread_failed` is the return status of a thread to report a failure. It is neither `NULL` nor does it denote a valid address.

Enhancement Orbix enhancement.

IT_ThreadBody Class

`IT_ThreadBody` is the base class for thread execution methods. To start a thread, derive a class from `IT_ThreadBody`, add any data members needed by the thread, and provide a `run()` method which does the thread's work. Then use an `IT_ThreadFactory` object to start a thread that will execute the `run()` method of your `IT_ThreadBody` object.

If a derived `IT_ThreadBody` contains data, then it must not be destroyed while threads are using it. One way to manage this is to allocate the `IT_ThreadBody` with the `new()` operator and have the `IT_ThreadBody` delete itself at the end of `run()`. Also, if multiple threads run the same `IT_ThreadBody`, it is up to you to provide synchronization on shared data.

```
class IT_ThreadBody {
public:
    virtual ~IT_ThreadBody() {}

    virtual void* run() =0;
};
```

IT_ThreadBody::~IT_ThreadBody() Destructor

```
virtual ~IT_ThreadBody();
```

The destructor for the `IT_ThreadBody` object.

IT_ThreadBody::run()

```
virtual void* run() =0;
```

Does the work and returns a status, which is typically `NULL` or the address of a static object.

Exceptions

On platforms that support exceptions, if `run()` throws an exception while used by an attached thread, this thread's exit status will be `IT_Thread::thread_failed`.

IT_ThreadFactory Class

An `IT_ThreadFactory` object starts threads that share some common properties. You can derive your own class from `IT_ThreadFactory` to control other aspect of thread creation, such as the exact method used to create or start the thread, or the priority of threads when they are created.

```
class IT_ThreadFactory {
public:
    enum DetachState { Detached, Attached };

    IT_ThreadFactory(
        DetachState detachState,
        size_t stackSize =0
    );

    virtual ~IT_ThreadFactory();

    virtual IT_Thread start(
        IT_ThreadBody& body,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    static IT_Thread smf_start(
        IT_ThreadBody& body,
        DetachState detach_state,
        size_t stack_size,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

protected:
...
```

IT_ThreadFactory::DetachState Enumeration

```
enum DetachState { Detached, Attached };
```

A thread can be started in a detached or attached state. If a thread is detached, you cannot join it (retrieve its exit status). If a thread is attached you must join it to tell the operating system to forget about it.

Enhancement Orbix enhancement.

IT_ThreadFactory::IT_ThreadFactory() Constructor

```
IT_ThreadFactory(
    DetachState detachState,
    size_t stackSize = 0
);
```

Constructor for an `IT_ThreadFactory` object.

Parameters

`detachState`	Specify whether the manufactured threads are `Detached` or `Attached`.
`stackSize`	As an option, you can specify the stack size of your threads (expressed in bytes). A value of `0` (the default) means that the operating system will use a default.

Enhancement Orbix enhancement.

See Also `IT_Thread::join()`

IT_ThreadFactory::~IT_ThreadFactory() Destructor

```
virtual ~IT_ThreadFactory();
```

The destructor for a thread factory object.

Enhancement Orbix enhancement.

IT_ThreadFactory::smf_start()

```
static IT_Thread smf_start(
    IT_ThreadBody& body,
    DetachState detach_state,
    size_t stack_size,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

A static member method (smf) that starts a thread without creating a thread factory explicitly. This method is useful for simple examples and prototyping but is not as flexible for robust applications.

Enhancement Orbix enhancement.

See Also IT_ThreadFactory::start()

IT_ThreadFactory::start()

```
virtual IT_Thread start(
    IT_ThreadBody& body,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Starts a thread. This method creates an operating system thread that runs the given body. The method returns an IT_Thread object that represents this thread.

Parameters

body	The thread body to run.
eh	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions An error that can be reported includes IT_TSRuntime.

See Also IT_Thread
IT_ThreadBody

IT_TimedCountByNSemaphore Class

This semaphore is a non-negative counter typically used to coordinate access to a set of resources. Several resources can be posted or waited for atomically. For example, if there are 5 resources available, a thread that asks for 7 resources would wait but another thread that later asks for 3 resources would succeed, taking 3 resources.

```
class IT_TimedCountByNSemaphore {
   public:
    enum { infinite_timeout = -1 };
    enum { infinite_size = 0 };

    IT_TimedCountByNSemaphore(
        size_t initial_count,
        size_t max_size,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    ~IT_TimedCountByNSemaphore();

    void post(
        size_t n,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void wait(
        size_t n,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    int wait(
        size_t n,
        long timeout,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    int trywait(
        size_t n,
```

```
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

private:
...
};
```

IT_TimedCountByNSemaphore::infinite_size Constant

```
enum { infinite_size = 0 };
```

A constant used to indicate an infinite sized semaphore.

See Also IT_TimedCountByNSemaphore::wait()

IT_TimedCountByNSemaphore::infinite_timeout Constant

```
enum { infinite_timeout = -1 };
```

A constant used to indicate there is no time-out period for the semaphore.

See Also IT_TimedCountByNSemaphore::wait()

IT_TimedCountByNSemaphore:: IT_TimedCountByNSemaphore() Constructor

```
IT_TimedCountByNSemaphore(
    size_t initial_count,
    size_t max_size,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Initializes the semaphore with `initial_count` and sets its maximum size to `max_size`.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is IT_TSRuntime.

IT_TimedCountByNSemaphore:: ~IT_TimedCountByNSemaphore() Destructor

```
~IT_TimedCountByNSemaphore();
```

The destructor for the semaphore.

Enhancement Orbix enhancement.

IT_TimedCountByNSemaphore::post()

```
void post(
    size_t n,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Posts the number of resources managed.

Parameters

`n`	The number of resources. If the value of `n` plus the previous number of resources is greater than `max_size`, then the number of resources remains unchanged and an `IT_TSLogic` error is reported. Calling the method using a value of 0 does nothing.
`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

`IT_TSRuntime`
`IT_TSLogic`

IT_TimedCountByNSemaphore::trywait()

```
int trywait(
    size_t n,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Equivalent to a `wait(n, 0, eh)`.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is IT_TSRuntime.

See Also IT_TimedCountByNSemaphore::wait()

IT_TimedCountByNSemaphore::wait()

```
void wait(
    size_t n,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Attempts to take a set of resources atomically.

```
int wait(
    size_t n,
    long timeout,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Attempts to take a set of resources (n) atomically. Returns 1 upon success or 0 when the operation times out. Calling wait(0, timeout, eh) returns 1 immediately.

Parameters

n	The number of resources attempted. A value of 0 causes the methods to return immediately.
timeout	The number of milliseconds before the call gives up. You can use the constant infinite_timeout.
eh	A reference to an error handler object.

IT_Semaphore and IT_TimedSemaphore can be more efficient than IT_TimedCountByNSemaphore when resources are posted and waited for one by one.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is IT_TSRuntime.

See Also IT_Semaphore
IT_TimedSemaphore

IT_TimedOneshot Class

An `IT_TimedOneshot` class is a synchronization policy typically used to establish a rendezvous between two threads. It can have three states:

- RESET
- SIGNALED
- WAIT

The key class members are as follows:

```
class IT_TimedOneshot {
public:
    enum { infinite_timeout = -1 };

    IT_TimedOneshot(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    ~IT_TimedOneshot();

    void signal(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void reset(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void wait(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    int wait(
        long timeout,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    int trywait(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
```

```
...
};
```

IT_TimedOneshot::infinite_timeout Constant

```
enum { infinite_timeout = -1 };
```

The `IT_TimedOneshot` class includes the symbolic constant `infinite_timeout`. This constant has a value of -1.

Enhancement Orbix enhancement.

See Also `IT_TimedOneshot::wait()`

IT_TimedOneshot::IT_TimedOneshot() Constructor

```
IT_TimedOneshot(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Initializes the one-shot to the `RESET` state.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_TimedOneshot::~IT_TimedOneshot() Destructor

```
~IT_TimedOneshot();
```

Destroys the one-shot object.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_TimedOneshot::reset()

```
void reset(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Resets the one-shot object.

- Resetting a one-shot while in the SIGNALED state changes its state to RESET.
- Resetting a one-shot while in the RESET state has no effect.
- Resetting a one-shot in the WAIT state is an error. Note that this error is not always detected, even in the checked mode.

Parameters

eh	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_TimedOneshot::signal()

```
void signal(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Signals the one-shot.

- Signaling a one-shot while in the RESET state changes its state to SIGNALED.
- Signaling a one-shot while in the WAIT state atomically releases the waiting thread and changes the one-shot state to RESET.
- Signaling a one-shot while in the SIGNALED state is an error.

Parameters

eh	A reference to an error handler object.

Enhancement Orbix enhancement.

IT_TimedOneshot::trywait()

```
int trywait(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
```

```
);
```

Equivalent to a call to `wait(0, eh)`.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

See Also IT_TimedOneshot::wait()

IT_TimedOneshot::wait()

```
void wait(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);

int wait(
    long timeout,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Waits for the one-shot.

- Waiting for a one-shot while in the `RESET` state changes its state to `WAIT`. the second method returns `1` when another thread signals the one-shot within the time-out period. Otherwise it returns `0` and changes the state back to `RESET`.
- Waiting for a one-shot while in the `SIGNALED` state changes its state to `RESET`. The first method returns immediately and the second method returns 1 immediately.
- Waiting for a one-shot while in the `WAIT` state is an error.

Parameters

`timeout`	The number of milliseconds before the call gives up. You can use the constant `infinite_timeout`.
`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

See Also IT_Semaphore
IT_TimedSemaphore

IT_TimedSemaphore Class

The `IT_TimedSemaphore` object is a counter with a timer for coordinating access to some resources.

```
class IT_TS_API IT_TimedSemaphore
{
public:
    enum { infinite_timeout = -1 };

    IT_TimedSemaphore(
        size_t initial_count,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    ~IT_TimedSemaphore();

    void post(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    void wait(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
    int wait(
        long timeout,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    int trywait(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );
private:
...
};
```

IT_TimedSemaphore::infinite_timeout Constant

```
enum { infinite_timeout = -1 };
```

The `IT_TimedSemaphore` class includes the symbolic constant `infinite_timeout`. This constant has a value of -1.

Enhancement Orbix enhancement.

See Also `IT_TimedSemaphore::wait()`

IT_TimedSemaphore::IT_TimedSemaphore() Constructor

```
IT_TimedSemaphore(
    size_t initial_count,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

A semaphore constructor.

Parameters

`initial_count`	Initializes the semaphore's counter with this value.
`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is `IT_TSRuntime`.

IT_TimedSemaphore::~IT_TimedSemaphore() Destructor

```
~IT_TimedSemaphore();
```

The destructor.

Enhancement Orbix enhancement.

IT_TimedSemaphore::post()

```
void post(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is `IT_TSRuntime`.

IT_TimedSemaphore::trywait()

```
int trywait(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Returns 1 if a resource has been obtained, 0 otherwise.

Parameters

`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is `IT_TSRuntime`.

IT_TimedSemaphore::wait()

```
void wait()
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);

int wait(
    long timeout,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Waits for one resource. The `wait()` method blocks if the semaphore's counter value is 0 and decrements the counter if the counter's value is greater than 0.

Parameters

`timeout`	The number of milliseconds before the call gives up. You can also use the constant `infinite_timeout`.
`eh`	A reference to an error handler object.

Enhancement Orbix enhancement.

Exceptions Errors that can be reported include:

IT_TSRuntime
IT_TSLogic

IT_TSBadAlloc Error Class

When `new()` returns `0` an `IT_TSBadAlloc` exception is reported.

```
class IT_TS_API IT_TSBadAlloc : public IT_TSRuntime
public:
    IT_TSBadAlloc();
    virtual ~IT_TSBadAlloc();
    virtual void raise() const;
};
```

See Also

IT_TSRuntime
IT_TSError

IT_TSError Error Class

All errors reported by the TS package are `IT_TSError` objects. The key members of the class are as follows:

```
class IT_TS_API IT_TSError {
public:
    IT_TSError(
        unsigned long TS_errcode,
        long OS_errno = 0
    );
    IT_TSError(
        const IT_TSError& other
    );

    virtual ~IT_TSError();

    unsigned long TS_error_code() const;
    long OS_error_number() const;
    const char* what() const;
    virtual void raise() const;

protected:
...
```

See Also IT_DefaultTSErrorHandler

IT_TSError::IT_TSError() Constructors

```
IT_TSError(
    unsigned long TS_errcode,
    long OS_errno = 0
);

IT_TSError(
    const IT_TSError& other
);
```

Constructs an error with this TS error code and optionally an error number given by the operating system. The second method is the copy constructor.

Enhancement Orbix enhancement.

IT_TSError::~IT_TSError() Destructor

```
virtual ~IT_TSError();
```

The destructor.

Enhancement Orbix enhancement.

IT_TSError::OS_error_number()

```
long OS_error_number() const;
```

Returns the operating system error number that represent the error. Returns 0 if the error is not reported by the operating system.

Enhancement Orbix enhancement.

IT_TSError::raise()

```
virtual void raise() const;
```

When exceptions are supported, this method throws `*this`, a pointer to this `IT_TSError` object. If exceptions are not supported, it calls `::abort()`.

Enhancement Orbix enhancement.

IT_TSError::TS_error_code()

```
unsigned long TS_error_code() const;
```

Returns the TS error code that represents the error.

Enhancement Orbix enhancement.

IT_TSError::what()

```
const char* what();
```

Returns a string describing the error. The caller must not de-allocate the returned string.

Enhancement Orbix enhancement.

See Also IT_TSLogic
IT_TSRuntime
IT_TSBadAlloc

IT_TSErrorHandler Class

The last parameter of almost every TS method is a reference to an object of the class `IT_TSErrorHandler`. When a TS method detects an error, it creates an `IT_TSError` object and passes it to `IT_TSErrorHandler::handle()`.

```
class IT_TS_API IT_TSErrorHandler {
public:
    virtual ~IT_TSErrorHandler();

    virtual void handle(
        const IT_TSError& thisError
    ) = 0;
};
```

See Also `IT_DefaultTSErrorHandler`

IT_TSErrorHandler::handle()

```
virtual void handle(
   const IT_TSError& thisError
) = 0;
```

Handles the given TS error.

Parameters

`thisError`	The error raised.

Enhancement Orbix enhancement.

IT_TSErrorHandler::~IT_TSErrorHandler() Destructor

```
virtual ~IT_TSErrorHandler();
```

The destructor for the error handler object.

Enhancement Orbix enhancement.

IT_TSLogic Error Class

An `IT_TSLogic` error signals an error in the application's logic, for example when a thread attempts to join itself.

```
class IT_TS_API IT_TSLogic : public IT_TSError {
   public:
    IT_TSLogic(
        unsigned long code,
        long fromOS =0
    );

    virtual ~IT_TSLogic();

   virtual void raise() const;

private:
// ...
};
```

See Also

IT_TSError
IT_TSRuntime

IT_TSRuntime Error Class

An `IT_TSRuntime` error is an error detected by the operating system or by the underlying thread package.

```
class IT_TS_API IT_TSRuntime : public IT_TSError {
public:
    IT_TSRuntime(
        unsigned long code,
        long fromOS =0
    );

    virtual ~IT_TSRuntime();

    virtual void raise() const;

private:
...
```

See Also

IT_TSError
IT_TSRuntime

IT_TSVoidStar Class

An `IT_TSVoidStar` object is a data entry point that can be shared by multiple threads. Each thread can use this entry point to get and set a `void*` pointer that refers to thread-specific (private) data.

```
class IT_TSVoidStar {
public:
    IT_TSVoidStar(
        void (*destructor)(void*) df,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

    ~IT_TSVoidStar();

    void* get(
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    ) const;

    void set(
        void* newValue,
        IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
    );

private:
...
};
```

IT_TSVoidStar::IT_TSVoidStar() Constructor

```
IT_TSVoidStar(
    void (*destructor)(void*) df,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Constructs an `IT_TSVoidStar` object. Initially, all thread-specific pointers are `NULL`.

Parameters

`df`	You can optionally associate a non-NULL destructor method with an `IT_TSVoidStar` object. Before exiting, a thread will call this destructor with its specific pointer value only when its specific pointer value is not `NULL`.
`eh`	A reference to an error handler object.

On some platforms, when threads are not started using an `IT_ThreadBody`, the application might have to call explicitly `IT_CurrentThread::cleanup()` upon thread exit to perform this cleanup.

Enhancement Orbix enhancement.

Exceptions An error that can be reported is `IT_TSRuntime`.

See Also `IT_TSVoidStar::~IT_TSVoidStar()`
`IT_CurrentThread::cleanup()`

IT_TSVoidStar::~IT_TSVoidStar() Destructor

```
~IT_TSVoidStar();
```

The destructor for an `IT_TSVoidStar` object.

If a non-`NULL` destructor method is associated with this `IT_TSVoidStar` object (by way of the `IT_TSVoidStar()` constructor), and the thread-specific value of this object is not `NULL`, the non-NULL destructor method is called with the thread-specific value.

WARNING: If the `IT_TSVoidStar` object has a non-NULL destructor, do not destroy the object while any other threads have a non-NULL thread-specific pointer. This is because on some platforms, a newly allocated `IT_TSVoidStar` object might *reincarnate* the destroyed `IT_TSVoidStar` object and its thread-specific values. This can lead to unexpected results.

Enhancement Orbix enhancement.

See Also `IT_TSVoidStar::IT_TSVoidStar()`

IT_TSVoidStar::get()

```
void* get(
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
) const;
```

Gets the pointer associated with the calling thread. Returns `NULL` when the calling thread did not explicitly set this value.

Exceptions An error that can be reported is `IT_TSRuntime`.

Enhancement Orbix enhancement.

IT_TSVoidStar::set()

```
void set(
    void* newValue,
    IT_TSErrorHandler& eh IT_TS_DEFAULT_ERROR_HANDLER
);
```

Sets the pointer associated with the calling thread to `newValue`.

Exceptions An error that can be reported is `IT_TSRuntime`.

Enhancement Orbix enhancement.

Index