PROGRESS®
ORBIX®

Internationalization Guide

Version 6.3.5, July 2011

PROGRESS software
BUSINESS MAKING PROGRESS™

Third Party Acknowledgements:

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: 13-Jul-2011

Contents

List of Tables

Preface

Audience

This guide is intended for Orbix programmers who develop products that might be internationalized.This guide also provides information to administrators of Orbix deployments where internationalization is required.

Related documentation

The Orbix documentation set includes the following related documentation:

- *CORBA Programmer's Guide* (C++ and Java)
- *CORBA Programmer's Reference* (C++ and Java)
- *Orbix Administrator's Guide*

The latest versions of all Orbix documentation can be found online at `http://www.iona.com/support/docs`.

Additional resources

The IONA knowledge base (http://www.iona.com/support/knowledge_base/index.xml) contains helpful articles, written by IONA experts, about the Orbix and other products.

The IONA update center (http://www.iona.com/support/updates/index.xml) contains the latest releases and patches for IONA products.

If you need help with this or any other IONA products, contact IONA at `support@iona.com`. Comments on IONA documentation can be sent to `docs-support@iona.com`.

Typographical conventions

This guide uses the following typographical conventions:

`Constant width`

Constant width (courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, and data structures. For example, text might refer to the `CORBA::Object` class.

Constant width paragraphs represent code examples or information a system displays on the screen. For example:

```
#include <stdio.h>
```

Italic

Italic words in normal text represent *emphasis* and *new terms*.

Italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example:

```
% cd /users/your_name
```

Note: Some command examples may use angle brackets to represent variable values you must supply. This is an older convention that is replaced with *italic* words or characters.

Keying conventions

This guide may use the following keying conventions:

No prompt

When a command's format is the same for multiple platforms, a prompt is not used.

`%`

A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.

`#`

A number sign represents the UNIX command shell prompt for a command that requires root privileges.

`>`

The notation > represents the DOS, Windows NT, Windows 95, or Windows 98 command prompt.

```
...
.
.
.
```

Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.

[] Brackets enclose optional items in format and syntax descriptions.

{} Braces enclose a list from which you must choose an item in format and syntax descriptions.

| A vertical bar separates items in a list of choices enclosed in {} (braces) in format and syntax descriptions.

CHAPTER 1

Orbix Internationalization

Orbix includes features that enable development and deployment of applications that manipulate user data encoded in characters beyond the traditional ASCII and ISO 8859-1 (Latin 1) code sets.

In this chapter

This chapter discusses the following topics:

Code Sets

Overview

A *coded character set*, or *code set* for short, is a mapping between integer values and characters they represent. The best known code set is ASCII, which defines 94 graphic characters and 34 control characters using the 7-bit integer range.

European languages

The 94 characters defined by the ASCII code set are sufficient for English, but they are not sufficient for European languages, such as French, Spanish, and German.

To remedy the situation, an 8-bit code set, ISO 8859-1, also known as Latin-1, was invented. The lower 7-bit portion is identical to ASCII. The extra characters in the upper 8-bit range cover those languages used widely in the Western European region.

Many other code sets are defined under ISO 8859 framework. These cover languages in other regions of Europe as well as Russian, Arabic and Hebrew. The most recent addition is ISO 8859-15, which is a revision of ISO 8859-1 and it adds the Euro currency symbol and other letters while removing less used characters. For further information about ISO-8859-x encoding, refer to the Web site “The ISO 8859 Alphabet Soup”.

Ideograms

Asian countries that use ideograms in their writing systems needed more characters than they fit in an 8-bit integer. Therefore, they invented a double-byte code set, where a character is represented by a bit pattern of 2 bytes.

These languages also needed to mix the double-byte code set with ASCII in a single text file. So, *character encoding schema*, or simply *encodings*, was invented as a way to mix characters of multiple code sets.

Some of the popular encodings used in Japan include:

- Shift JIS
- Japanese EUC
- Japanese ISO 2022

Unicode

Unicode is a new code set that is gaining popularity. It aims to assign a unique number, or code point, to every character that exists (and even once existed) in all languages. To accomplish this, Unicode, which began as a double-byte code set, has been expanded into a quadruple-byte code set.

Unicode, in pure form, can be difficult to use within existing computer architectures, because many APIs are byte-oriented and assume that the byte value 0 means the end of the string.

For this reason, Unicode Transformation Format for 8-bit channel, or UTF-8, is frequently used. When browsers list “Unicode” in its encoding selection menu, they usually mean UTF-8, rather than the pure form of Unicode.

Visit Unicode Inc. for more information about Unicode and its variants.

Charset names

To address the need for computer networks to connect different types of computers that use different encodings, the Internet Assigned Number Authority, or IANA, has a registry of encodings at http://www.iana.org/assignments/character-sets.

IANA names are used by many Internet standards including MIME, HTML, and XML.

Table 1 lists IANA names for some popular charsets.

Table 1: *IANA Charset Names*

IANA Name	Description
US-ASCII	7-bit ASCII for US English
ISO-8859-1	Western European languages
UTF-8	Byte oriented transformation of Unicode
UTF-16	Double-byte oriented transformation of 4-byte Unicode
Shift_JIS	Japanese DOS & Windows
EUC-JP	Japanese adaptation of generic EUC scheme, used in Unix
ISO-2022-JP	Japanese adaptation of generic ISO 2022 encoding scheme

Note: IANA names are case insensitive. For example, US-ASCII can be spelled as us-ascii or US-ascii.

CORBA Names

In CORBA code sets are identified by numerical values registered with the Open Group's registry, OSF Code Set Registry: ftp://ftp.opengroup.org/pub/code_set_registry/code_set_registry1.2g.txt.

Java Names

Java has its own names for charsets. For example, ISO-8859-1 is named `ISO8859_1`, Shift_JIS is named `SJIS`, and UTF-8 is named `UTF8`.

Java is transitioning to IANA charset names, to be aligned with MIME. JDK 1.3 and above recognizes both names.

Note: This guide uses IANA charset names even for CORBA code sets.

Locales

Concept of locale

Most of modern operating systems are multilingual. Users can choose their language, and the operating system behaves according to the linguistic convention of the chosen language. For example, the user the number 1234.56 can be displayed as `1,234.56` to the English users and `1.234,56` to the German users.

However, language alone is not enough to determine a behavior, especially for languages that are used in many countries. For example, French speakers in Canada expect to see `1,234.56` while European French speakers expect to see `1.234,56`.

The concept of *locale* addresses these issues. A locale combines charsets and display behavior for specific regions.

ISO standards

The International Standard Organization, ISO, defines two standards to specify locale. In general, ISO 639 specifies the language code and ISO 3166 specifies the country code.

These standards can be down loaded from the ISO web site at `www.iso.org`.

Operating system locales

Windows

On Windows, the **Regional and Language Option** control panel is used to select a locale. Locales are listed by language. For languages that are spoken in multiple countries, such as English, you must choose the setting for your region. Regions are listed next to the language in parenthesis. For example a Spanish speaker in Mexico would select **Spanish (Mexico)**.

UNIX

On Solaris and other POSIX conformant platforms, a set of environment variables, `LANG` and `LC_xxxx`, are used to select a language. Usually, only `LANG` is used to set locale behavior to the specified locale.

Typical locale values on Solaris are `en` (English, generic), `en_GB` (British English), or `en_GB.ISO8859-15` (British English using `ISO-8859-15` encoding). The two-letter code, `en`, means English, and is taken from the language code standard from ISO, and GB means Great Britain and is taken from another ISO standard on the country code. As the last example indicates, an OS locale name can also include an encoding name (ISO8859-15).

Note: The name of the encoding does not follow IANA charset registry.

When no encoding name appears as a component of the locale name, a default encoding is implied. For example, the locale ja on Solaris implies use of EUC-JP encoding, eucJP in the Solaris naming convention. Therefore, ja and ja_JP.eucJP are synonymous on Solaris.

All Unix-derived operating systems share similar locale semantics although the naming conventions vary widely.

Java locales

Java has its own locale mechanism. Its notational convention is similar to Solaris, except that there is no encoding specifier as Java's internal encoding is always UTF-16. Typical locales include:

- en_US
- fr_FR
- de_DE
- zh_CN
- zh_TW
- ja_JP
- ko_KR

Java’s default locale is automatically inherited from the JVM’s current operating system locale. Example 1 shows how to use the Java Locale class to determine the locale and file encoding settings for your system.

Example 1: *Printing the Java locale and file encoding setting*

```
// Java
import java.util.Locale;

public class printLocale
{
  public static void main(String [])
  {
    Locale default_locale = Locale.getDefault();
    System.out.println("Default locale: " +
   default_locale.toString());
```

Example 1: *Printing the Java locale and file encoding setting*

```
    String file_encoding = System.getProperty("file.encoding");
    System.out.println("File encoding: " + file_encoding);
  }
}
```

For more information, see the JDK API document for the `Locale` class.

Language on the Internet

The Hyper Text Transfer Protocol, HTTP, has two headers, `Accept-Language` and `Content-Language,` that relate to locale. These two headers take a language value (or list of values) which has the form `*language*[-*subtag*].`

According the HTTP specification, `*language*` and `*subtag*` can be any string value. In practice, the ISO two-letter language code and country code are used. Therefore, the values in these two fields are almost same as the Java locales.

Orbix Internationalization

Feature list

The following internationalization features are available in Orbix:

- CORBA Internationalization Features

  This includes IDL `wchar` and `wstring` datatypes, code set negotiation, and extended code set support.

- Java 2 Enterprise Edition (J2EE) 1.3

  This includes the servlet `response.setContentType()` and `request.setCharacterEncoding()` methods and JSP `pageEncoding` and `contentType` attributes in the page directive.

- Locale-to-encoding mapping enhancement to J2EE

  An encoding can be associated with a locale via configuration. When a JSP or Servlet specifies a locale attribute for a response using the `response.setLocale()` method, the encoding associated with the locale is used in the charset component of the `Content-Type` header of the response.

- Configurable fall-back encoding of servlet for J2EE

  A fall-back encoding for requests on a servlet or JSP can be configured using the URL mapping for the servlet. If the servlet does not associate an encoding with a request programmatically (for example, using `request.setContentType()`) then the fall-back encoding is used, if configured.

- IANA-charset-name to Java-converter-name mapping for J2EE

  In a small number of cases it is necessary to decode HTTP request body data using a Java converter with a different name from the name of the IANA charset used in the request header. ASP provides a mechanism to map IANA charset names to Java converter names via configuration.

- Character encoding support in WSDL Test Client

  WSDL Test Client has a menu to specify the encoding
- Large number of code sets/character encoding supported.

  163 built-in code sets are supported for CORBA. All character encodings (code sets) supported by the underlying JDK can be used by J2EE programs.

Enabling tools

With Orbix, you can enable applications from different locales or using different code sets to interoperate. However, Orbix does not provide tools to help you applications capable of working with multiple locales or code sets.

CHAPTER 2

CORBA Internationalization

Orbix lets you run applications in numerous locales.

In this chapter

This chapter discusses the following topics:

Overview

Wide characters

CORBA 2.1 introduced the datatypes `wchar` (*wide character*) and `wstring` (*wide string*). Wide characters allow a character to be stored in a fixed length datatype whether it is a one byte character, a double-byte character, or a quad-byte character. This makes programming for multiple languages easier.

Note: The actual size of a wide character varies by operating system. Typical sizes are two bytes or four bytes.

Example 2 shows an IDL definition that uses wide datatypes.

Example 2: *Sample IDL using wide datatypes*

```
// IDL
interface WideEcho
{
  wstring echo(in wstring ws);
  wstring echoSingleWChar(in wchar wc);
};
```

Mixing wide and narrow data

The traditional `string` datatype, sometimes called a *narrow* string, can also represent a multibyte character string. A character in this form has a varying byte length, usually 1 to 3 bytes. The number of bytes depends on the code set in use.

The `char` datatype, however, cannot store a multibyte character because its size is limited to one byte. Example 3 shows an IDL definition of an interface that mixes narrow and wide strings.

Example 3: *Mixing narrow and wide strings*

```
// IDL
interface WideEcho
{
  wstring echo(in wstring ws);
  wstring echoSingleWChar(in wchar wc);
  wstring echoNarrowString(in string ns);
};
```

Code set negotiation

Because CORBA is designed to work in a heterogeneous networking environment, the server’s native code set might differ from the client’s native code set. CORBA defines a mechanism for ensuring that both client and server can exchange meaningful data efficiently. This process is called *code set negotiation*.

Supported Code Sets

Popular code sets

Table 2 shows some of the code sets that Orbix supports.

Table 2: *Popular code sets supported by Orbix*

OSF code set name	OSF code set id	IANA charset	Java encoding
ISO 8859-1:1987	0x00010001	ISO-8859-1	ISO8859_1
UCS-2, Level 1	0x00010100	UCS-2	UTF-16
UCS-4, Level 1	0x00010104	UCS-4	UCS-4
X/Open UTF-8	0x05010001	UTF-8	UTF8
JIS eucJP	0x00030010	EUC-JP	EUC_JP
OSF Japanese SJIS-1	0x05000011	Shift_JIS	SJIS

Java CORBA

For Java CORBA, Orbix supports 163 code sets. It uses the Java native encoding converters listed for JDK 1.4 (http://java.sun.com/j2se/1.4/docs/guide/intl/encoding.doc.html) and for JDK 1.3 (http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html).

Because Java does not use OSF code set IDs to name the encodings, an OSF code set ID must first be mapped to a Java encoding name.

Because the mapping between OSF code set IDs and IANA and Java naming schema is not one-to-one, Orbix maps the most popular code set among the code sets that are almost identical. For example, Orbix supports code set id `0x05000011` (OSF Japanese SJIS-1) but not `0x05020002` (JVC_SJIS). Please make sure to use the code set that Orbix supports.

WARNING: The mapping is subject to change without notice in future releases.

One of the most popular code sets used in Japan, `ISO-2022-JP`, is missing from OSF registry therefore Orbix does not support it. `ISO-2022-JP` is mainly used in e-mail, and it is rarely used in inter-process communication or in storage.

C/C++ CORBA

For C and C++ CORBA, Orbix supports 112 code sets. It uses the code set converters from ICU (http://oss.software.ibm.com/icu), an open-source project supported by IBM.

Custom code set plugins

The Java encoders and the ICU encoders built in to Orbix do not support all code sets in use today. For situations where conversion is needed for an unsupported code set, Orbix has a plugable code set converter architecture which makes it possible to write and add a custom code set converter plugin.

IONA Professional Service can write a custom code set plugin for you. Contact your local IONA sales office for details.

Code Set Negotiation

Overview

Code set negotiation is the process by which two CORBA processes which use different *native code set*s determine which code set to use as a *transmission code set*. Occasionally, the process requires the selection of a *conversion code set* to transmit data between the two processes. The algorithm is defined in section 13.10.2.6 of the CORBA specification (http://cgi.omg.org/docs/formal/02-12-06.pdf).

Note: For CORBA programing in Java, you can specify codeset other than the true native codeset.

Native code set

A native code set (NCS) is a code set that a CORBA program speaks natively. For Java, this is UTF-8 (`0x05010001`) for `char` and `String`, and UTF-16(`0x00010109`) for `wchar` and `wstring`. For C and C++, this is the encoding that is set by `setlocale()`, which in turn depends on the `LANG` and `LC_xxxx` environment variables.

Conversion code set

A conversion code set (CCS) is an alternative code set that the application registers with the ORB. More than one CCS can be registered for each of the narrow and wide interfaces. CCS should be chosen so that the expected input data can be converted to and from the native code set without data loss. For example, Windows code page 1252 (`0x100204e4`) can be a conversion code set for ISO-8859-1 (`0x00010001`), assuming only the common characters between the two code sets are used in the data.

Each application has its own native code set and a set of conversion code sets for `char` and `string`. Each application also has a separate native code set and conversion code sets for `wchar` and `wstring`. The CCS for `wchar` and `wstring` can be same as or different from those for `char` and `string`.

Transmission code set

A transmission code set (TCS) is the code set agreed upon after the code set negotiation. The data on the wire uses this code set. It will be either the native code set, one of the conversion code sets, or UTF-8 for the narrow interface and UTF-16 for the wide interface.

Negotiation algorithm

Code set negotiation uses the following algorithm to determine which code set to use in transferring data between client and server:

1. If the client and server are using the same native code set, no translation is required.
2. If the client has a converter to the server's code set, the server's native code set is used as the transmission code set.
3. If the client does not have an appropriate converter and the server does have a converter to the client's code set, the client's native code set is used as the transmission code set.
4. If neither the client nor the server has an appropriate converter, the server ORB tries to find a conversion code set that both server and client can convert to and from without loss of data. The selected conversion code set is used as the transmission code set.
5. If no conversion code set can be found, the server ORB determines if using UTF-8 (narrow characters) or UTF-16 (wide characters) will allow communication between the client and server without loss of data. If UTF-8 or UTF-16 is acceptable, it is used as the transmission code set. If not, a `CODESET_INCOMPATIBLE` exception is raised.

Code set compatibility

The last steps involves a compatibility test, but the CORBA specification does not define when a code set is compatible with another. The compatibility test algorithm employed in Orbix is outlined below:

1. ISO 8859 Latin-*n* code sets are compatible.
2. UCS-2 (double-byte Unicode), UCS-4 (four-byte Unicode), and UTF-x are compatible.
3. All other code sets are not compatible with any other code sets.

This compatibility algorithm is subject to change without notice in future releases. Therefore, it is best to configure the code set variables as explicitly as possible in order to reduce dependency on the compatibility algorithm.

Configuring the Code Set Plugin

Configuration variables

In order for an ORB to transmit character data in a code set other than ISO 8859-1, the ORB must be configured properly. Four configuration variables control the code set plugin:

plugins:codeset:char:ncs specifies the native code sets used to represent narrow characters.

plugins:codeset:char:ccs specifies the list of conversion code sets supported for narrow characters.

plugins:codeset:wchar:ncs specifies the native code sets used to represent wide characters.

plugins:codeset:wchar:ccs specifies the list of conversion code sets supported for wide characters.

For more information on these variables, see the *Orbix Configuration Reference*.

Note: For CORBA programing in Java, you can specify a codeset other than the true native codeset. Please see "Native code set" on page 23.

Light weight code set plugin

The standard code set plugin for C/C++ requires 8MB of memory. If memory is limited or if you are planning to use only the code sets listed in Table 3, a light weight replacement, `it_basic_codeset`, is available.

Table 3:*Code sets supported by the light weight plugin*

UCS-2
UTF-4
UTF-16
ISO-8859-1
EBCDIC

To use this plugin instead of the standard one, modify the configuration with this setting

```
initial_references:IT_CodeSet_Registry:plugin= "basic_codeset";
```

Note: Changing the code set plugin in this way has no effect on Java.

Choice of conversion code sets

When choosing code sets to use as conversion code sets, three points should be considered:

- Data compatibility

  Conversion code sets should be chosen to minimize the of loss or corruption of data. At minimum, the code sets chosen must cover all the characters that need to be handled by the application. Shift_JIS and EUC-JP are based on the same base character sets, except that EUC-JP includes an extra code set that is in rare use. If the application does not expect to handle the extra code set, these code sets can be treated as compatible. Usually, Shift_JIS and ISO-8859-1 are not considered compatible because many of the letters with diacritics in ISO-8859-1 do not exist in Shift_JIS. However, if the application is not expected to handle data that includes these incompatible characters, you can consider them compatible. Ultimately, it is up to the application designer to decide whether a code set is compatible with another.

- Performance

  The choice of the transmission code set greatly affects the performance. In general, fixed-length code sets such as ISO-8859-x, EBCDIC, UCS-2 and UCS-4 can achieve better performance than the variable-length code sets such as UTF-8, EUC-JP and Shift_JIS.

Note: Although UTF-16 is a variable length code set, Orbix implements it as fixed-width code set because the characters beyond Basic Multilingual Plain (BMP) are not supported.

- Compatibility with legacy CORBA services

  It is often the case that an application uses multiple CORBA servers. Some servers might not support code sets other than ISO-8859-1, which is the only code set that is mandated to be supported by CORBA. Some servers might also not support wchar or wstring. In fact, most of the IONA services such as naming, locator, etc. belong to this category. For this reason, ISO-8859-1 should be included in plugins:codeset:char:ccs.

Example configurations

Example 4 shows a basic configuration that works in a mixed Java/C++ environment. This configuration works in Latin-1 based locales.

Example 4: *Basic code set configuration*

```
plugins:codeset:char:ncs = "0x05010001"; # UTF-8
plugins:codeset:char:ccs = ["0x00010001"]; # ISO-8859-1;
plugins:codeset:wchar:ncs = "0x00010109"; # UTF-16
plugins:codeset:wchar:ccs = [];
```

Example 5 shows a configuration for a heterogeneous system environment where some servers are written in Java and not configured to accept ISO-8859-1, and some servers are hosted on a mainframe. Because EBCDIC (IBM code page 037) does not include characters needed for European languages, this configuration can only be used for English.

Example 5: *Mixed environment configuration*

```
plugins:codeset:char:ncs = "0x00010001"; # ISO-8859-1
plugins:codeset:char:ccs = ["0x05010001", "0x10020025"]; # UTF-8, EBCDIC(IBM-037)
plugins:codeset:wchar:ncs = "0x00010001"; # ISO-8859-1
plugins:codeset:wchar:ccs = ["0x00010109";]; # UTF-16
```

For the language such as Japanese, where multiple code sets are used in a heterogeneous system environment, all of the used code sets should be put in the conversion code set list. Example 6 shows a configuration for a Japanese system with servers running on both Windows and Solaris systems.

Example 6: *Japanese mixed environment configuration*

```
plugins:codeset:char:ncs = "0x05010001"; # UTF-8
plugins:codeset:char:ccs = ["0x00030010", "0x05000011", "0x00010001"]; # JIS eucJP,
   OSF SJIS, ISO-8859-1
plugins:codeset:wchar:ncs = "0x00010109"; # UTF-16
plugins:codeset:wchar:ccs = ["0x00030010", "0x05000011", "0x00010104", "0x00010100"];
   # JIS eucJP, OSF SJIS, UCS-4 Level 1, UCS-2 Level 1
```

Logging

The code set plugin outputs informational event messages using the event subsystem `IT_CODESET`. To view these events make sure the configuration variable `event_log:filters` includes the entry `"IT_CODESET=*"` or `"IT_CODESET=INFO"`.

Default configuration

If any code set configuration variables are missing from the configuration, the default values shown in the *Orbix Configuration Reference* are used.

Java Internationalization

IDL-to-Java mapping

As specified by the OMG's IDL-to-Java mapping specification, the `string` and `wstring` IDL datatypes are mapped to the Java `String` class, and the `char` and `wchar` IDL datatypes are mapped to the Java `char` datatype.

Note: Although a Java `char` is two-bytes wide, any attempt to transmit a Java `char` that is bigger than `0xff` through the IDL `char` interface will throw the exception of `org.omg.CORBA.DATA_CONVERSION`.

The latest version of the CORBA *IDL-to-Java Language Mapping Specification* can be obtained at http://www.omg.org/cgi-bin/doc?formal/02-08-05.pdf.

Coding

Because Java treats both narrow and wide IDL datatypes alike, there is little that a CORBA developer using Java needs to consider when thinking about internationalization.

To use `WideEcho.idl`, shown in Example 3 on page 12, apply `idlgen` using the following command:

```
idlgen java_poa_genie.tcl -jP WideEchoDemo -all WideEcho.idl
```

To implement the server, `WideEchoImpl.java` needs to have its operation defined. Each operation in the `WideEcho` class simply echoes the string that is passed into it. Example 7 shows the implementation of the `WideEcho` operations.

Example 7: *WideEcho Java server*

```
// Java
public java.lang.String echo(java.lang.String ws)
throws org.omg.CORBA.SystemException
{
 return ws;
}
```

Example 7: *WideEcho Java server*

```
public java.lang.String echoSingleWChar(char wc)
throws org.omg.CORBA.SystemException
{
  // Returns a String consisting of just one char.
  char [] x = new char[1];
  x[0] = wc;
  return new String(x);
}
public java.lang.String echoNarrowString(java.lang.String ns)
throws org.omg.CORBA.SystemException
{
  return ns;
}
```

The Java implementation of the IDL operation `wstring echo(wstring)` is identical to the Java implementation of the IDL operation `wstring echoNarrowString(string)`. This is because Java makes no distinction between an IDL `string` and IDL `wstring`.

The Java implementation of the IDL operation `wstring echoSingleWChar(char)` is more complicated than the others only because it needs to convert a Java `char` to a Java `String`.

On the client side, the code is similar to a traditional Java CORBA program. IDL `wchar` and IDL `char` are both represented as Java `char`. IDL `wstring` and IDL `string` are represented as Java `String`.

Native code set

For Java-written CORBA programs, the real native code set is always UTF-16, as mandated by the Java language specification. However, you can declare any code set as the native code set for the purpose of CORBA, as long as the code set supports the language you need to support.

One limitation applies: only byte-oriented code sets. code sets that do not include `null`, can be set to `plugins:codeset:char:ncs` and `plugins:codeset:char:ccs`. For `plugins:codeset:wchar:ncs` and `plugins:codeset:wchar:ccs`, any code set can be used, whether it is byte-oriented or not.

C/C++ Internationalization

IDL-to-C++ mapping

Although both C and C++ have different IDL mapping specifications, most CORBA platforms using C map wchar and wsring with the C++ mappings. Table 4 shows IDL to C++ mappings for both narrow and wide character data.

Table 4: *IDL to C++ character mappings*

IDL	C++
char	char
string	char *
wchar	wchar_t (CORBA::WChar)
wstring	wchar_t * (CORBA::WChar *)

The latest version of the CORBA *C Language Mapping Specification* can be obtained at http://www.omg.org/cgi-bin/doc?formal/99-07-35.pdf.

The latest version of the CORBA *C++ Language Mapping Specification* can be obtained at http://www.omg.org/cgi-bin/doc?formal/99-07-41.pdf.

Coding

Because C++ maps wide character data to special datatypes, coding internationalizable applications requires a few modifications:

- C and C++ programs must include the file locale.h. This file contains the definitions and headers used to support code sets and locale functionality.
- setlocale(LC_ALL, "") must be called in the application's main() routine. This operation determines the system's locale settings and initializes the appropriate code sets. The encoding of the char[] and wchar_t is determined by setlocale().
- The stream output operator << does not take wchar_t (IDL wchar) or wchar_t * (IDL wstring) properly when applied to cout. In order to print these datatypes, use the wide-oriented stream object wcout.

- idlgen generates a class named IT_GeniePrint which has two methods that appear to print wide data, print_wstring() and print_wchar(). However, these methods print all characters in the hex form \xNNNN and can only be used when a hex dump is needed during debug. All wide data must be converted before being printed.

Example

To implement the WideEcho.idl example, Example 3 on page 12, run idlgen as follows:

```
idlgen cpp_poa_genie.tcl -all WideEcho.idl
```

This generates a makefile for the Visual C++ nmake utility. To run the IDL compiler on Unix systems, run nmake.

On the server side, the operations echo(), echoSingleWchar() and echoNarrowString() must be implemented in WideEchoImpl.cxx. Example 8 shows an implementation of the operations.

Example 8: *C++ server implementation of WideEcho.idl*

```
// C++
#include <locale.h>
CORBA::WChar* WideEchoImpl::echo(const CORBA::WChar* ws)
IT_THROW_DECL((CORBA::SystemException))
{
  return CORBA::wstring_dup(ws);
}
CORBA::WChar* WideEchoImpl::echoSingleWChar(CORBA::WChar wc)
IT_THROW_DECL((CORBA::SystemException))
{
  wchar_t x[2];
  x[0] = wc;
  x[1] = (wchar_t) 0;
  return CORBA::wstring_dup(x);
}
```

Example 8: *C++ server implementation of WideEcho.idl*

```
CORBA::WChar* WideEchoImpl::echoNarrowString(const char* ns)
IT_THROW_DECL((CORBA::SystemException))
{
  CORBA::WChar* _result;

  int xlen = strlen(ns)+1; // Max len of buf needed.
  wchar_t *x = new wchar_t[xlen]; // Temp buffer
  mbstowcs(x, ns, xlen);
  _result = CORBA::wstring_dup(x);
  delete [] x; // Clean up temp buffer
  return _result;
}
```

Note: `CORBA::WChar` is equivalent to `wchar_t`. `idlgen` generates code using `CORBA::WChar`.

Because C++ maps `string` and `wstring` to different datatypes, the implementation of `echoNarrowString()` must explicitly convert `ns` from `char *` into `wchar_t *` using the ANSI C standard function `mbstowcs()`.

On the client side, in `client.cxx`, the calling convention is no different from the traditional CORBA convention. Example 9 shows a client implementation.

Example 9: *C++ client implementation for WideEcho.idl*

```
// C++
#include <locale.h>
main(int argv, char[] argc))
{
  CORBA::Object_var obj;

  setlocale(LC_ALL, "") // set the locale
 ...
  obj = read_reference("WideEcho.ref");
  WideEcho_var WideEcho1 = WideEcho::_narrow(obj);

...
// Replace Hello with your language equivalent
  wcout << WideEcho1->echo(L"Hello in wstring") << endl
  wcout << WideEcho1->echoSingleWChar(L"H") << endl
  wcout << WideEcho1->echoNarrowString("Hello in string") <<
    endl;
}
```

Native code set

The native code set of C and C++ applications is determined by the platform's locale setting. You must set the Orbix native code set to `plugins:codeset:char:ncs` and `plugins:codeset:wchar:ncs`.

On Windows in Western European locales, the native code set for the narrow char/string (NCS-C) is Windows Code Page 1252 which is ISO 8859-15. Since the OSF registry lacks Window Code Page 1252 or ISO 8859-15, you can use ISO 8859-1 as the best approximation and set the configuration variable as follows:

```
plugins:codeset:char:ncs = "0x00010001"; # ISO-8859-1
```

On WIndows in a Japanese locale, the NCS-C is Window Code Page 932, an extension of Shift_JIS. Because the OSF registry also lacks Window Code Page 932, you can use OSF SJIS as the closest approximation and set the configuration variable as follows:

```
plugins:codeset:char:ncs = "0x05000011"; # OSF SJIS
```

On WIndows, the native code set for the `wchar/wstring` (NCS-W) is always UCS-2 regardless of locale. Set the configuration variable as follows:

```
plugins:codeset:wchar:ncs = “0x00010100”; # UCS-2 Level 1
```

On Solaris, both NCS-C and NCS-W are determined by the current locale. For the ISO 8859-1 based locales such as `C`, `en`, `fr`, `de`, `es`, `it` and `pt`, both NCS-C and NCS-W should be set to `ISO-8859-1`. So, the configuration variables are set as follows:

```
plugins:codeset:char:ncs = "0x00010001"; # ISO-8859-1
plugins:codeset:wchar:ncs = "0x00010001"; # ISO-8859-1
```

For Solaris in the Japanese `ja` locale, both NCS-C and NCS-W should be set to the OSF code set equivalent of `EUC-JP` as follows:

```
plugins:codeset:char:ncs = "0x00030010"; # JIS eucJP
plugins:codeset:wchar:ncs = "0x00030010"; # JIS eucJP
```

For Solaris in UTF-8 based locales such as `en_US.UTF-8`, `ja_JP.UTF-8`, `ko_KR.UTF-8`, `zh_CN.UTF-8` and `zh_TW.UTF-8`, NCS-C is UTF-8 and NCS-W is UCS-4. Set the configuration variables as follows:

```
plugins:codeset:char:ncs = "0x05010001"; # UTF-8
plugins:codeset:wchar:ncs = "0x00010104"; # UCS-4 Level 1
```

CHAPTER 3

Restrictions

Orbix has some limitations in its internationalization support.

In this chapter

This chapter discusses the following topics:

Translations

Orbix is internationalized, but it is not localized. All the GUI applications and messages remain in English.

There are some exceptions. Some GUI elements and messages that originate from the underlying operating system or Java run-time environment are localized automatically. For example, the **OK** button is translated in some dialog boxes.

Similarly, some messages from the operating systems are in the language of the locale.

Property Values

Generally speaking, various properties that Orbix uses are restricted to the traditional ASCII range.

For example, the following properties must be in ASCII in order for IONA to guarantee their proper behavior:

- File path
- User ID
- Password
- URL
- Repository name
- Channel name
- Configuration domain name
- Configuration variable name
- Configuration variable value
- Scope name
- Cluster name
- (J2EE) Application name

The only Orbix property that can have non-ASCII characters is the role names if Orbix is configured to use an LDAP server that supports non-ASCII.

COMet

COMet supports non-ASCII characters in the same manner as regular ORBs using the IDL `string` and `char` datatypes. It does not support `wstring` or `wchar`.

Be careful that a multibyte character is not passed to a `char` parameter because a `char` is a single byte storage. Attempts to do so result in an abnormal program termination on the COMet side.

Glossary

A

administration

All aspects of installing, configuring, deploying, monitoring, and managing a system.

ART

Adaptive Runtime Technology. IONA's modular, distributed object architecture, which supports dynamic deployment and configuration of services and application code. ART provides the foundation for IONA software products.

C

CFR

See configuration repository.

client

An application (process) that typically runs on a desktop and requests services from other applications that often run on different machines (known as server processes). In CORBA, a client is a program that requests services from CORBA objects.

codeset

A *coded character set*, or *code set* for short, is a mapping between integer values and characters they represent. The best known code set is ASCII, which defines 94 graphic characters and 34 control characters using the 7-bit integer range.

codeset negotiation

Code set negotiation is the process by which two CORBA processes that use different *native code set*s determine which code set to use as a *transmission code set*. Occasionally, the process requires the selection of a *conversion code set* to transmit data between the two processes. The algorithm is defined in section 13.10.2.6 of the CORBA specification (http://cgi.omg.org/docs/formal/02-12-06.pdf)

configuration

A specific arrangement of system elements and settings.

configuration domain

Contains all the configuration information that Orbix ORBs, services and applications use. Defines a set of common configuration settings that specify available services and control ORB behavior. This information consists of configuration variables and their values. Configuration domain data can be implemented and maintained in a centralized Orbix configuration repository or as a set of files distributed among domain hosts. Configuration domains let you organize ORBs into manageable groups, thereby bringing scalability and ease of use to the largest environments. See also configuration file and configuration repository.

configuration file

A file that contains configuration information for Orbix components within a specific configuration domain. See also configuration domain.

configuration repository

A centralized store of configuration information for all Orbix components within a specific configuration domain. See also configuration domain.

configuration scope

Orbix configuration is divided into scopes. These are typically organized into a root scope and a hierarchy of nested scopes, the fully-qualified names of which map directly to ORB names. By organizing configuration properties into various scopes, different settings can be provided for individual ORBs, or common settings for groups of ORB. Orbix services, such as the naming service, have their own configuration scopes.

CORBA

Common Object Request Broker Architecture. An open standard that enables objects to communicate with one another regardless of what programming language they are written in, or what operating system they run on. The CORBA specification is produced and maintained by the OMG. See also OMG.

CORBA objects

Self-contained software entities that consist of both data and the procedures to manipulate that data. Can be implemented in any programming language that CORBA supports, such as C++ and Java.

D

deployment

The process of distributing a configuration or system element into an environment.

H

HTTP

HyperText Transfer Protocol. The underlying protocol used by the World Wide Web. It defines how files (text, graphic images, video, and other multimedia files) are formatted and transmitted. Also defines what actions Web servers and browsers should take in response to various commands. HTTP runs on top of TCP/IP.

I

IDL

Interface Definition Language. The CORBA standard declarative language that allows a programmer to define interfaces to CORBA objects. An IDL file defines the public API that CORBA objects expose in a server application. Clients use these interfaces to access server objects across a network. IDL interfaces are independent of operating systems and programming languages.

IIOP

Internet Inter-ORB Protocol. The CORBA standard messaging protocol, defined by the OMG, for communications between ORBs and distributed applications. IIOP is defined as a protocol layer above the transport layer, TCP/IP.

installation

The placement of software on a computer. Installation does not include configuration unless a default configuration is supplied.

Interface Definition Language

See IDL.

invocation

A request issued on an already active software component.

IOR

Interoperable Object Reference. See object reference.

L

locale

A locale combines charsets and display behavior for specific regions. Language alone is not enough to determine a behavior, especially for languages that are used in many countries. For example, French speakers in Canada expect to see `1,234.56` while European French speakers expect to see `1.234,56`. The concept of locale addresses these issues.

O

object reference

Uniquely identifies a local or remote object instance. Can be stored in a CORBA naming service, in a file or in a URL. The contact details that a client application uses to communicate with a CORBA object. Also known as interoperable object reference (IOR) or proxy.

OMG

Object Management Group. An open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications, including CORBA. See www.omg.com.

ORB

Object Request Broker. Manages the interaction between clients and servers, using the Internet Inter-ORB Protocol (IIOP). Enables clients to make requests and receive replies from servers in a distributed computer environment. Key component in CORBA.

P

POA

Portable Object Adapter. Maps object references to their concrete implementations in a server. Creates and manages object references to all objects used by an application, manages object state, and provides the infrastructure to support persistent objects and the portability of object implementations between different ORB products. Can be transient or persistent.

protocol

Format for the layout of messages sent over a network.

S

server

A program that provides services to clients. CORBA servers act as containers for CORBA objects, allowing clients to access those objects using IDL interfaces.

T

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic suite of protocols used to connect hosts to the Internet, intranets, and extranets.

Index