

WebFOCUS

National Language Support
for International Computing

Contents

1	Understanding National Language Support	1-1
	What is National Language Support (NLS)?.....	1-2
	NLS and NLV	1-2
	Understanding Code Pages.....	1-3
	Character Sets.....	1-3
	Defining Scripts, Languages, and Code Pages	1-4
	Code Points	1-6
	Code Pages in Client/Server Computing	1-7
	Information Builders' NLS API	1-8
	How the NLS API Works.....	1-9
	Multiple Client Code Pages on Different Platforms.....	1-10
	Reading Data in Different European Languages With a Single Code Page	1-12
	Combining Data From Different Code Pages With an EDA Hub Server.....	1-14
	Combining Data in Different Scripts With Unicode	1-16
2	Understanding National Language Versions	2-1
	What is a National Language Version (NLV)?	2-2
	National Language Version Components.....	2-2
	Installing National Language Versions	2-3
	Using National Language Versions.....	2-4
	Local Language Online Help.....	2-7
	Working With an Unfamiliar Local Language.....	2-8
	Interpreting Local Language Error Messages.....	2-11
	Changing the Local Language.....	2-12
3	Automated Configuration of NLS for Windows NT and UNIX.....	3-1
	Basic Initialization of Windows NT or UNIX Server NLS Services	3-2
	Basic Initialization of Windows NT or UNIX Client NLS Services	3-9
4	Custom Configuration of NLS for Windows NT and UNIX.....	4-1
	Customizing Your Windows NT or UNIX Server for NLS Services.....	4-2
	Changing Your Windows NT or UNIX Server Code Page Settings	4-4
	Identifying Current Windows NT or UNIX Code Page Settings (Step 1 of 5)	4-4
	Identifying Alternate or Additional Windows NT or UNIX Code Pages (Step 2 of 5).....	4-5
	Adding Alternate or Additional Windows NT or UNIX Code Pages (Step 3 of 5).....	4-8
	Generating Windows NT or UNIX Transcoding Tables Using the TSGU (Step 4 of 5)	4-8
	Changing the Windows NT or UNIX Server Code Page Setting (Step 5 of 5).....	4-9
	Configuring Windows NT or UNIX Server NLS Default Characteristics	4-9

- Configuring Customized Windows NT or UNIX Server NLS Monocasing4-12
- Configuring for Customized Windows NT or UNIX Server NLS Sort Sequences4-14
- Testing Your Widows NT or UNIX Server NLS Behavior.....4-17
- Customizing Windows NT or UNIX Client NLS Services4-17
 - Changing Your Windows NT or UNIX Client Code Page Setting4-17
- 5 Configuring NLS for MVS.....5-1**
 - Configuring Your MVS Server for NLS Services.....5-2
 - Changing Your MVS Server Code Page Settings5-4
 - Identifying Current MVS Code Page Settings (Step 1 of 5).....5-4
 - Identifying Alternate or Additional MVS Code Pages (Step 2 of 5).....5-5
 - Adding Alternate or Additional MVS Code Pages (Step 3 of 5).....5-8
 - Generating MVS Transcoding Tables Using the TSGU (Step 4 of 5)5-8
 - Changing the MVS Server Code Page Setting (Step 5 of 5)5-9
 - Configuring MVS Server NLS Default Characteristics.....5-9
 - Configuring Customized MVS Server NLS Monocasing5-12
 - Configuring for Customized MVS Server NLS Sort Sequences5-14
 - Modifying the Server JCL5-17
 - Customizing MVS Client NLS Services5-17
 - Changing Your MVS Client Code Page Setting5-17
- 6 Configuring NLS for WebFOCUS (Windows Version)6-1**
 - NLS Architecture6-2
 - ASCII-Transparency6-2
 - Data Interpretation.....6-4
 - Default FOCShell Code Page Type6-5
 - Default EDA Server Code Page Type6-7
 - DOS and Windows Code Page Setting6-8
 - Using the NLS Configuration Utility to Configure for Local Usage.....6-9
 - Installation Considerations6-9
 - Configuring for Client/Server Usage.....6-13
 - Default Client/Server Configuration6-14
 - Configuring for Single-Byte Character Languages Accessing ANSI (Windows) Data6-14
 - Configuring for Single-Byte Character Languages Accessing OEM (DOS) Data6-15
 - Configuring for Double-Byte Character Languages.....6-15
 - Special Code Page Considerations6-16
 - Use of Code Page Parameter in Master Files6-16
 - Displaying NLS Information: ? LANG.....6-17

7	Unicode	7-1
	What is Unicode?	7-2
	Why Use Unicode?.....	7-2
	Unicode at Information Builders	7-3
	Configuring Your System for Unicode.....	7-3
	Determining if Unicode is Necessary	7-4
	Defining Alphanumerics for Unicode Data Type.....	7-5
	Understanding Your Code Page Environment	7-5
8	Troubleshooting	8-1
	Troubleshooting Strategy	8-2
	Questions to Ask	8-2
	Observing Results	8-3
	Additional Resources	8-3
	Identifying NLS Issues	8-4
	NLS Configuration Issues	8-5
	Reviewing NLS Configuration Settings.....	8-9
	Code Page Consistency	8-9
	Display Issues.....	8-10
	Printing Issues	8-10
	Technical Support.....	8-11
A	TSGU and NLS Configuration File	A-1
	Generating New Transcoding Tables	A-2
	The NLS Configuration File.....	A-4
B	History of Code Pages	B-1
	Origins of Code Pages	B-2
	Far Eastern Encodings	B-3
	Unicode	B-4
	Guidelines to Selecting a Code Page	B-4
C	Euro Currency and NLS SET Commands	C-1
	Integrating the Euro Currency	C-2
	Converting Currencies.....	C-2
	Preparing for Currency Conversions	C-4
	Creating the Currency Data Source.....	C-4
	Identifying Fields That Contain Currency Data	C-8
	Activating the Currency Data Source	C-10
	Querying the Currency Data Source in Effect	C-11

Contents

Processing Currency Data	C-12
Punctuating Large Numbers	C-16
Accessing Local Language Error Messages	C-17
D Bibliography.....	D-1
Glossary	G-1
Index	I-1

EDA, FIDEL, FOCUS, FOCUS Fusion, Information Builders, the Information Builders logo, TableTalk, and Web390 are registered trademarks and Parlay, SiteAnalyzer, SmartMart, WebFOCUS, and WorldMART are trademarks of Information Builders, Inc.

Acrobat and Adobe are registered trademarks of Adobe Systems Incorporated.

Allaire and JRun are trademarks of Allaire Corporation.

NOMAD is a registered trademark of Aonix.

UniVerse is a registered trademark of Ardent Software, Inc.

AvantGo is a trademark of AvantGo, Inc.

WebLogic is a registered trademark of BEA Systems, Inc.

SUPRA and TOTAL are registered trademarks of Cincom Systems, Inc.

Alpha, DEC, DECnet, and NonStop are registered trademarks and Tru64, OpenVMS, and VMS are trademarks of Compaq Computer Corporation.

CA-ACF2, CA-Datcom, CA-IDMS, CA-Top Secret, and Ingres are registered trademarks of Computer Associates International, Inc.

MODEL 204 and M204 are registered trademarks of Computer Corporation of America.

Paradox is a registered trademark of Corel Corporation.

StorHouse is a registered trademark of FileTek, Inc.

HP MPE/iX is a registered trademark of Hewlett Packard Corporation.

Informix is a registered trademark of Informix Software, Inc.

Intel is a registered trademark of Intel Corporation.

ACF/VTAM, AIX, AS/400, CICS, DB2, DRDA, Distributed Relational Database Architecture, IBM, MQSeries, MVS, OS/2, OS/390, OS/400, RACF, RS/6000, S/390, VM/ESA, and VTAM are registered trademarks and DB2/2, HiperSpace, IMS, MVS/ESA, QMF, SQL/DS, VM/XA and WebSphere are trademarks of International Business Machines Corporation.

INTERSOLVE and Q+E are registered trademarks of INTERSOLVE.

Orbit is a registered trademark of Iona Technologies Inc.

Approach and DataLens are registered trademarks of Lotus Development Corporation.

ObjectView is a trademark of Matesys Corporation.

ActiveX, FrontPage, Microsoft, MS-DOS, PowerPoint, Visual Basic, Visual C++, Visual FoxPro, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Motorola is a trademark of Motorola, Inc.

Teradata is a registered trademark of NCR International, Inc.

Netscape, Netscape FastTrack Server, and Netscape Navigator are registered trademarks of Netscape Communications Corporation.

Nextel is a registered trademark of Nextel Communications.

Oracle is a registered trademark and Rdb is a trademark of Oracle Corporation.

Palm is a trademark and Palm OS is a registered trademark of Palm Inc.

PeopleSoft is a registered trademark of PeopleSoft, Inc.

BlackBerry is a trademark and RIM is a registered trademark of Research In Motion Limited.

Red Brick Warehouse is a trademark of Red Brick Systems.

SAP and SAP R/3 are registered trademarks and SAP Business Information Warehouse and SAP BW are trademarks of SAP AG.

ADABAS is a registered trademark of Software A.G.

CONNECT:Direct is a trademark of Sterling Commerce.

Java and all Java-based marks, NetDynamics, Solaris, SunOS, and iPlanet are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

PowerBuilder and Sybase are registered trademarks and SQL Server is a trademark of Sybase, Inc.

Unicode is a trademark of Unicode, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2001, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

Preface

This documentation provides an introduction to National Language Support (NLS). It includes an overview of NLS and instructions for installing and configuring NLS for Windows NT®, UNIX®, WebFOCUS (Windows® version), and MVS®. It is intended for anyone who wants to become more familiar with NLS concepts and architecture. It is particularly useful for the person responsible for planning the enterprise's NLS environment and architecture.

How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	<i>Understanding National Language Support</i>	Familiarizes you with the National Language Support (NLS) concepts and features. It describes code page architecture in client/server computing and the implementation of NLS at Information Builders. It also contains scenarios and diagrams that walk you through four possible NLS client and server configurations.
2	<i>Understanding National Language Versions</i>	Provides an overview of National Language Versions (NLVs).
3	<i>Automated Configuration of NLS for Windows NT and UNIX</i>	Describes the steps for installing and configuring the NLS component for Windows NT and UNIX for both client and server. It details the basic configuration of NLS using the standardized EDA configuration tool.
4	<i>Custom Configuration of NLS for Windows NT and UNIX</i>	Describes the steps for customizing the NLS component for Windows NT and UNIX for both client and server. Customization includes changing your code page settings, changing default NLS characteristics, custom monocasing, and custom sorting.
5	<i>Configuring NLS for MVS</i>	Describes the steps for configuring NLS for MVS for both client and server. It includes information on changing your code page settings, changing default NLS characteristics, custom monocasing, and custom sorting.
6	<i>Configuring NLS for WebFOCUS (Windows Version)</i>	Describes the architecture and installation instructions for the configuration of NLS for WebFOCUS (Windows version).

Chapter/Appendix		Contents
7	<i>Unicode</i>	Provides an overview of Unicode™. It also describes WebFOCUS support for reading Unicode data sources from the Report Assistant, Graph Assistant, and other Java™-based tools in WebFOCUS.
8	<i>Troubleshooting</i>	Discusses techniques for determining NLS problems and identifying resolutions
A	<i>TSGU and NLS Configuration File</i>	Describes the options available with the Transcoding Services Generation Utility (TSGU) and the NLS configuration file.
B	<i>History of Code Pages</i>	Discusses the history of code pages and the various encoding strategies developed for Far Eastern languages. It also describes the implementation of Unicode as a code page architecture and presents guidelines to assist you in your code page selection.
C	<i>Euro Currency and NLS SET Commands</i>	Describes the NLS SET commands for euro currency, punctuating large numbers, and local language error messages. In addition, it describes how to create and use a currency data source to convert to and from the new euro currency.
D	<i>Bibliography</i>	Provides additional information and reference material for NLS.
	<i>Glossary</i>	Provides definitions of key NLS and industry-related terms and concepts.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
<code>THIS TYPEFACE</code> or <code>this typeface</code>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable) in a text paragraph, indicates a cross-reference, or emphasizes an important term.
this typeface	Highlights file names and commands (in a text paragraph) that must be lowercase.
this typeface	Indicates buttons, menu items, and dialog box options you can click or select.
{ }	Indicates two choices from which you must choose one. You type one of these choices, not the braces.
[]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the information within the brackets, not the brackets.
	Separates two mutually exclusive choices in a syntax line. You type one of these choices, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameters, not the ellipsis points (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

See the *Information Builders Technical Publications Catalog* for the most up-to-date listing and prices of technical publications, plus ordering information. To obtain a catalog, contact the Publications Order Department at (800) 969-4636.

You can also visit our World Wide Web site, <http://www.informationbuilders.com>, to view a current listing of our publications and to place an order.

Customer Support

Do you have questions about NLS?

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your NLS questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code number (*xxxx.xx*).
- Your configuration:
 - The front-end you are using, including vendor and release.
 - The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
 - The software release.
 - The server you are accessing, including release (for example, 4.3.1).
- Your National Language Support (NLS) configuration:
 - The code page for both the client and server.
- The stored procedure (preferably with line numbers), FOCUS commands, or SQL statements being used.
- The name of the Master File and Access File.

- The exact nature of the problem:
 - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
 - The error message and return code, if applicable.
 - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing the code to access a single data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?
- Your EDA configuration:
 - The EDA Server(s) software version, release, and PUT level.
 - The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
 - The EDA Client software release.
- Your National Language Support (NLS) configuration:
 - The code page for both the client and server.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The database server release level.
- The database name and release level.
- The Master File and Access File.

User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Document Enhancement Request Form on our Web site, <http://www.informationbuilders.com>.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

CHAPTER 1

Understanding National Language Support

Topics:

- What is National Language Support (NLS)?
- NLS and NLV
- Understanding Code Pages
- Code Pages in Client/Server Computing
- Information Builders' NLS API

National Language Support (NLS) provides customization of software to enhance usability in a country or language. For Information Builders' products, the most important part of NLS is to seamlessly interpret all national characters embedded in any data source. NLS also includes correct sorting, upper/lower case conversion, and providing locally expected data formats (time, date, currency, and numbers).

A proprietary NLS API code page engine enables Information Builders' products to work in all countries and for all languages. The NLS API's main function is managing the transcoding of data between the client and server components, each of which may have different code pages. The NLS API also handles sorting, case conversion, and formatting dates, currency and numbers.

What is National Language Support (NLS)?

National Language Support (NLS) involves reading and interpreting data stored in code pages that represent the character sets of various international languages. For systems implemented across multiple platforms, NLS transcodes each graphic character in one code page into the corresponding graphic character in another code page. For example, to transcode the German graphic character 'Ä' from the IBM mainframe EBCDIC Germany code page 273, to its corresponding value in the PC Windows 1252 Western European code page, the NLS transcoding component changes hexadecimal value '4A' (mainframe) to the hexadecimal value 'C4' (Windows). For more information on code pages, see *Understanding Code Pages* on page 1-3.

Additional NLS functions include:

- **Sorting.** A sort order can be defined and additional letters from code pages can be added to accommodate sorting for specific local languages.
- **Monocasing.** Monocasing (also called case conversion) converts a letter from its lowercase to uppercase form (or vice versa).
- **Date and time formats.** Date and time formats can be selected for local languages.
- **Decimal notation and large number formatting.** Proper decimal notation and large number formatting for numbers and currency can be selected for local languages.

NLS and NLV

National Language Support (NLS) and National Language Version (NLV) are frequently confused. NLS ensures that systems can handle local language data. NLV is a version of a software product in which the entire user interface is displayed in a particular language. The user interface includes the following:

- Initial program installation
- Menus, toolbars, dialog boxes, forms
- Program utilities (assistants, wizards, editors)
- Online Help, including context-sensitive Help
- Error messages

Examples of NLV products at Information Builders are the French, German, Japanese, and Spanish editions of WebFOCUS. Information Builders' NLV products include both the translated and localized user interface, and handle local language data as well. For more information on NLVs, see Chapter 2, *Understanding National Language Versions*.

Understanding Code Pages

A code page is a matching set of numerical values and the written symbols they represent. Transcoding data between the client and server is the actual mapping of the graphic characters in one code page into the corresponding graphic characters of another code page.

Code pages have the following characteristics:

- Code pages are language specific and platform specific.
- Within a code page, a code point (numeric value) represents a specific written symbol.
- Each code page has a unique identifying number, which is also the name of the code page.
- Code page families are sets of code pages for a given platform.
- Several hundred code pages are used in different hardware, software, and data communications implementations worldwide.

Character Sets

There are two important types of computer character sets used to represent text: Single-Byte Character Set (SBCS) and Double-Byte Character Set (DBCS).

- **Single-Byte Character Set (SBCS)** code pages are 8-bit encodings used to represent scripts such as Eastern and Western European alphabets, Greek, Cyrillic (Russian), Arabic, Hebrew, and Thai.
- **Double-Byte Character Set (DBCS)** code pages use 16-bits to represent each written symbol. DBCS code pages are used for the East Asian (Chinese, Japanese, and Korean) scripts that have thousands of written symbols.

Defining Scripts, Languages, and Code Pages

Scripts, languages, and code pages are closely related in text handling for computer systems. This section will give you a better understanding of the differences in these terms.

A script is a collection of symbols used to represent textual information in a writing system. These symbols might be letters of the alphabet, the numerals 0-9, punctuation marks, and mathematical symbols. Scripts are the world's major writing systems, which include:

- Latin (Roman letters)
- Greek
- Cyrillic (Russian)
- Japanese

Written languages use symbols from a script to transcribe the spoken language. Languages with strong linguistic or historical links often make use of the same script. For example, most European languages use the Latin script, as does English. However, European languages have additional letters, referred to as national characters, which are not found in English. Examples of these national characters are German umlauts (Ä/ä, Ö/ö, Ü/ü), French accented characters (á, à, â), and Swedish national characters (Å/å, Ä/ä, Ö/ö).

In a similar way, Japanese makes extensive use of kanji, the Japanese forms of Chinese characters. However, many Chinese characters are not part of written Japanese, and some Japanese kanji are not found in the Chinese written language.

A code page assigns numeric values to a set of written symbols. Historically, the first code pages were for a single country or language. Recently, code pages have been designed to handle many languages using the same script. Examples of multi-language code pages include the almost identical Microsoft Windows 1252 and UNIX ISO 8859-1 code pages. These pages support almost all North American, South American, and Western European languages that use the Latin script.

In keeping with the trend toward designing code pages for multiple languages, Information Builders has developed its own proprietary code page 137, which handles all major North American, South American, and Western European languages (except Greek) for Windows NT and UNIX. Code page 137 is functionally equivalent to Microsoft Windows 1252 and UNIX ISO 8869-1 code pages.

Reference Information Builders' Key Code Pages

The following table describes the key code pages used by Information Builders' products:

Language	Windows NT	UNIX	MVS VM AS400	OS/390 (Open Edition)	Open VMS
English	437 (default)	437 (default)	37 (default)	37 (default)	437 (default)
Western European	137	137	Code page is dependent on country.	1047	NA
Central European	912	912	Code page is dependent on country.	1047	NA
Turkish	920	920	1026	1047	NA
Korean	949	949	933	933	949
Traditional Chinese	10948	10948	937	937	10948
Japanese	942	10942/942	939/930	939	10942
Hebrew	916	916	424	NA	NA
Unicode	65001	65001	NA	NA	65001

Note:

- MVS, VM, and AS/400 all use an IBM operating system and share the same family of EBCDIC code pages.
- NA indicates that the code page is currently unavailable but may be added if necessary.

Code Points

A code point is the numeric value assigned to a specific written symbol in a code page. For example, in the Windows 1252 code page shown below, the capital letter 'A' has a hexadecimal value of 0x41 where 0x is a prefix indicating a hex number and 41 are hex digits. Square boxes represent code points that have not been assigned a symbol.

The Windows 1252 and nearly identical UNIX ISO 8859-1 code pages contain all of the commonly used symbols for North America, South America, and Western Europe.

Windows 1252 code page

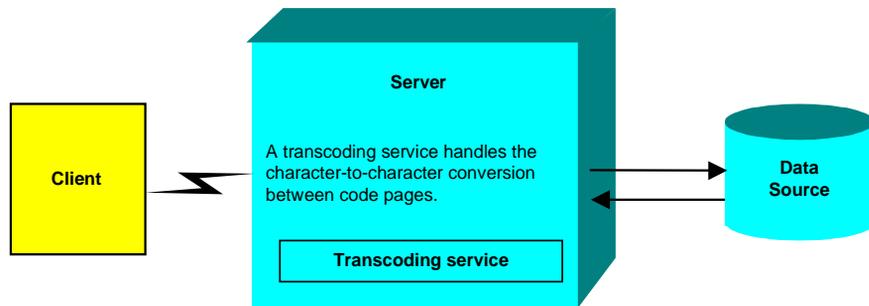
	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0		0	@	P	`	p	€	☐	NBSP	°	À	Ð	à	ø	
Code point 0x41 →	1	!	1	A	Q	a	q	☐	'	ı	±	Á	Ñ	á	ñ
2	"	2	B	R	b	r	,	'	¢	²	Â	Ò	â	ò	
3	#	3	C	S	c	s	f	"	£	³	Ã	Ó	ã	ó	
4	\$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô	
5	%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å	õ	
6	&	6	F	V	f	v	†	–	!	¶	Æ	Ö	æ	ö	
7	'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷	
8	(8	H	X	h	x	^	~	¨	,	È	Ø	è	ø	
9)	9	I	Y	i	y	‰	™	©	'	É	Ù	é	ù	
A	*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú	
B	+	;	K	[k	{	<	>	«	»	Ë	Û	ë	û	
C	,	<	L	\	l		Œ	œ	¬	¼	Ì	Ü	ì	ü	
Unassigned code point →	D	-	=	M	J	m	}	☐	☐	-	½	Í	Ý	í	ý
E	.	>	N	^	n	~	Ž	ž	®	¾	Î	Þ	î	þ	
F	/	?	O	_	o		☐	ÿ	—	¿	Ï	ß	ï	ÿ	

Code Pages in Client/Server Computing

Client/server computing products, such as WebFOCUS and EDA, generally have the client and server software installed on different computers. These computers can have different operating system platforms. Since codes pages are platform specific, the client and server components will use different code pages, even when processing the same language.

If client and server are using different code pages, a transcoding service must convert the binary values for each letter from one code page to the other. For example, a transcoding service must take the binary value of the letter 'A' on the server code page, and change it to a different binary value for of the letter 'A' on the client code page.

The following diagram illustrates the generic code page architecture for client/server computing:



Note: When referring to the Information Builders family of products, server refers specifically to an EDA Server. Client can refer to any of the following components connecting to an EDA Server:

- The WebFOCUS CGI or ISAPI client.
- WebFOCUS (Windows version).
- A third party product.

Information Builders' NLS API

The NLS API is Information Builders' proprietary facility for supporting international computing. Its primary function is the transcoding of data on a character-by-character basis, between the code page of the server component and the code page of the client component. The NLS API is a core component of EDA and is built into WebFOCUS and most other Information Builders' products.

In addition to the primary function of transcoding, the NLS API has the following additional features:

- **Sorting.** Sorting allows you to create rules for a sort order and add additional letters from code pages. For example, you may want to add German or Swedish umlauts (Ü) to your sort order. The sorting process can be configured to choose where you want the umlauts to fall in the sort sequence. For example, you may want the umlauts to fall at the end of the alphabet.
- **Monocasing.** Monocasing (also called case conversion) is the conversion of a letter from its lowercase to uppercase form (or vice versa). For example, in Standard French, all accented vowels lose their accents in the transition from lowercase to uppercase. Therefore, é, è, ê and e, all have the same uppercase character of E.

Note: The NLS API provides standard sorting and monocasing tables for supported code pages. Although rarely required, these tables can be customized after installation.

Information Builders' products have other localization features in addition to the NLS API. These features include the following:

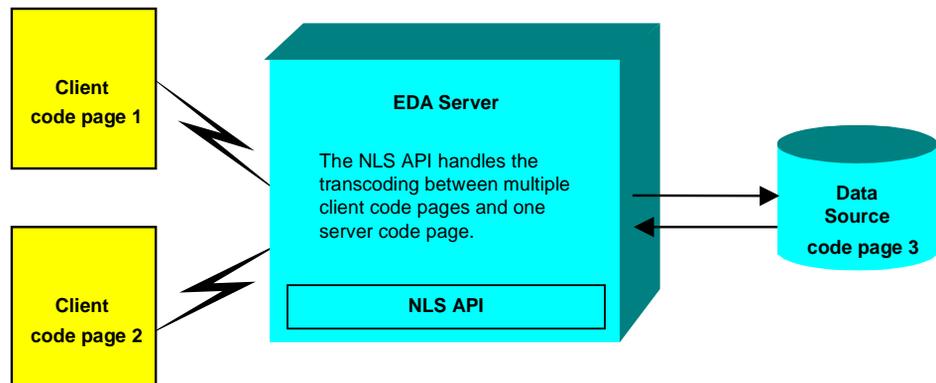
- **Decimal notation.** Countries differ in how they punctuate numbers. In the United States, numbers contain a decimal point and use a comma as a thousands separator. In other countries, a dot (period) or a blank is used as a thousands separator. Information Builders' products have a SET command that enables you to select the correct decimal notation for the selected language. For more information, see Appendix C, *Euro Currency and NLS SET Commands*.
- **Date and time capabilities.** Since there are numerous date and time formats used throughout the world, the NLS API enables you to choose date and time formats for the selected language. For example, the United States uses the 12-hour clock as a standard, while most of Europe uses the 24-hour clock. Similarly, the United States uses the date format MM/DD/YY while Britain uses DD/MM/YY. For additional information on the Data Description Language and Master Files, see the *Describing Data* manual.

How the NLS API Works

The NLS API code page architecture is Information Builders' proprietary implementation of the standard way of handling code pages used by all other client/server and database systems. A server can have one server code page for reading data sources but can support multiple client code pages. Server and client code pages are typically set during the installation and configuration process. During installation, the administrator selects one code page for the server. Based on this server code page selection, the NLS API generates the corresponding client code page settings, which are stored on the server. The initial NLS API settings can be modified by manually editing NLS configuration files after installation.

During installation, the server also generates a set of transcoding tables, which are resident in memory at run-time. As the number of different client code pages supported by the server increases, the number of tables defined by this transcoding service also increases. The Transcoding Services Generation Utility (TSGU), in addition to a configuration file containing a list of code pages, is used to generate the code page transcoding table. For details on configuring your server and client for NLS, refer to the installation and configuration documentation for your specific platform.

The following diagram illustrates the transcoding process between client and server:



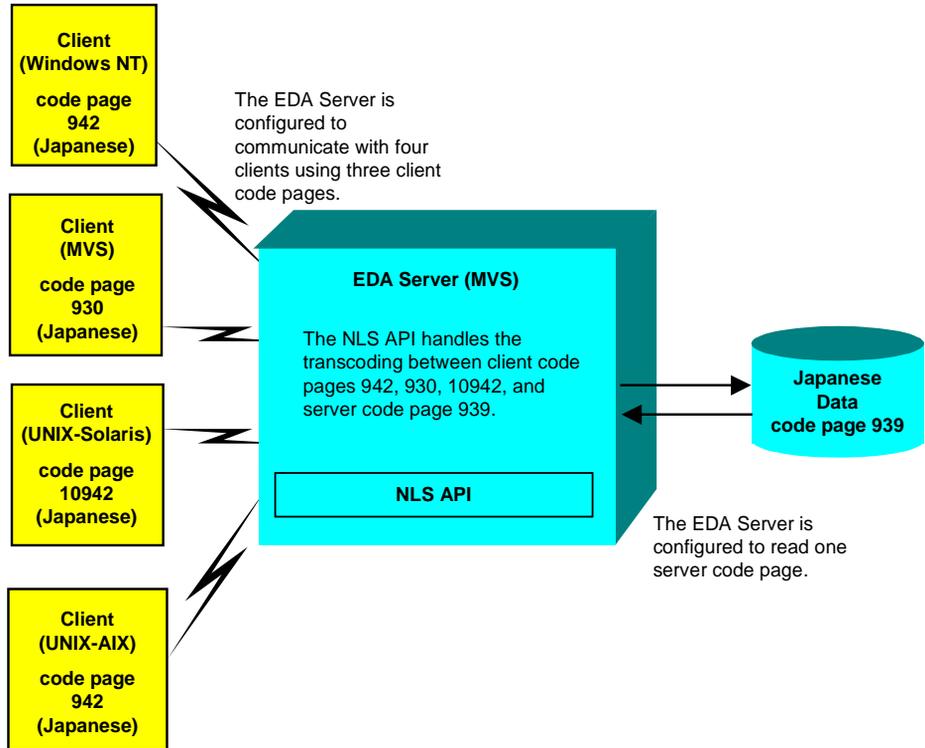
The remainder of this section presents four possible NLS client and server configurations. Each scenario consists of a description followed by a diagram.

Multiple Client Code Pages on Different Platforms

This scenario illustrates how the NLS API service supports transcoding with multiple client code pages and one data source code page all representing the same language.

Note: All code pages in this scenario represent Japanese however, the same transcoding takes place between the various platform specific code pages of any language.

1. The EDA Server is configured to communicate with three client code pages. All EDA Servers can support multiple client code pages.
2. The EDA Server is configured to read data with one server code page. All EDA Servers can have only one server code page setting.
3. A request from the client is passed through the NLS API and transcoded into the server code page.
4. Data from the data source is passed through the NLS API and transcoded into the appropriate client code page for the client making the request.
5. The result is the transparent transcoding of local language characters for all four clients and one data source.

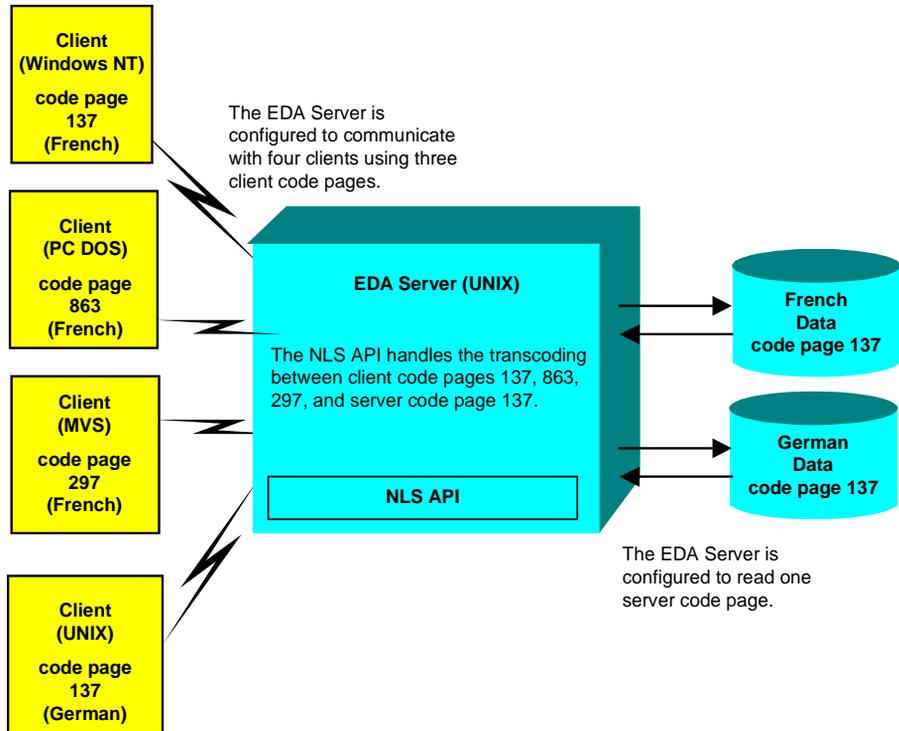


Reading Data in Different European Languages With a Single Code Page

In the following example, the two data sources are in different languages (French and German), but can be read with the same server code page (code page 137). Information Builders' code page 137 is based on the Latin script and it contains all symbols for languages of North America, South America, and Western Europe. It is implemented on both the Windows NT and UNIX platforms. For related information on the Latin script, see *Defining Scripts, Languages, and Code Pages* on page 1-4.

1. Both data sources use code page 137. If the two data sources have a common index key with a common range of values, they can be joined together for reporting. Otherwise, separate requests can be made against each data source.
2. The EDA Server is configured to communicate with three client code pages. All EDA Servers can support multiple client code pages.
3. A request from a client is passed through the NLS API on the EDA Server and transcoded into the server code page.
4. Data from the source file is passed through the NLS API and transcoded into the appropriate client code page for the client making the request.
5. This solution works only when data sources use the same code page. In this scenario, code page 137 handles French and German characters and is available on both the Windows NT and UNIX platforms. If data sources require different code pages, an EDA Hub Server is needed to combine them.

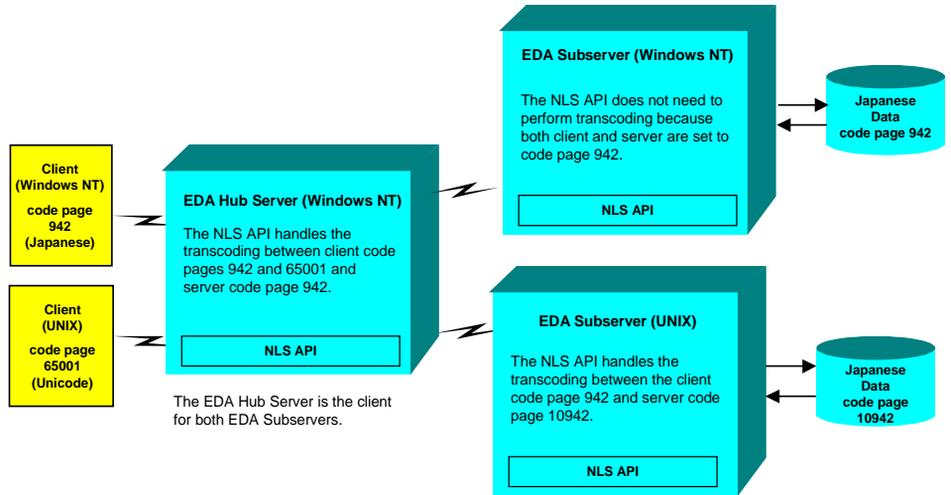
For details on combining different code pages with an EDA Hub Server, see *Combining Data From Different Code Pages With an EDA Hub Server* on page 1-14.



Combining Data From Different Code Pages With an EDA Hub Server

Data sources using different code pages for the same or different languages can be joined using an EDA Hub Server along with Subservers. The following is an overview of how transcoding is handled when combining data from different code pages.

1. The two Japanese data sources have a common index key with a common range of values to join them. One data source uses the Shift-JIS code page 942, and the other data source uses the EUC code page 10942.
 - a. Shift-JIS is PC data, therefore its EDA Subserver is running on a Windows NT Server.
 - b. EUC is UNIX data, therefore its EDA Subserver is running on a UNIX Server.
2. While the two EDA Subservers have different server code page settings, both are configured to communicate with the same client code page (Shift-JIS code page 942). The EDA Hub Server is the client for both EDA Subservers. Therefore, the server code page setting on the EDA Hub Server (Shift-JIS code page 942) is the same as the client code page setting for the EDA Subservers (Shift-JIS code page 942).
3. The EDA Hub Server has been configured to communicate with two client code pages (Shift-JIS code page 942 and UTF-8 Unicode code page 65001).
4. The Windows NT client could be a Web browser that is configured to read Shift-JIS, so it reads Japanese data from code page 942. The UNIX client could be the WebFOCUS CGI that is configured to read UTF-8 Unicode, so it reads Japanese data from code page 65001.
5. This solution only works when both data source code pages support the same script, in this case Japanese. You cannot combine Japanese and Chinese data in this way because Japanese and Chinese are different scripts and always use different code pages. To combine unrelated scripts, you have to use a Unicode solution. For information on combining unrelated scripts using Unicode, see *Combining Data in Different Scripts With Unicode* on page 1-16.



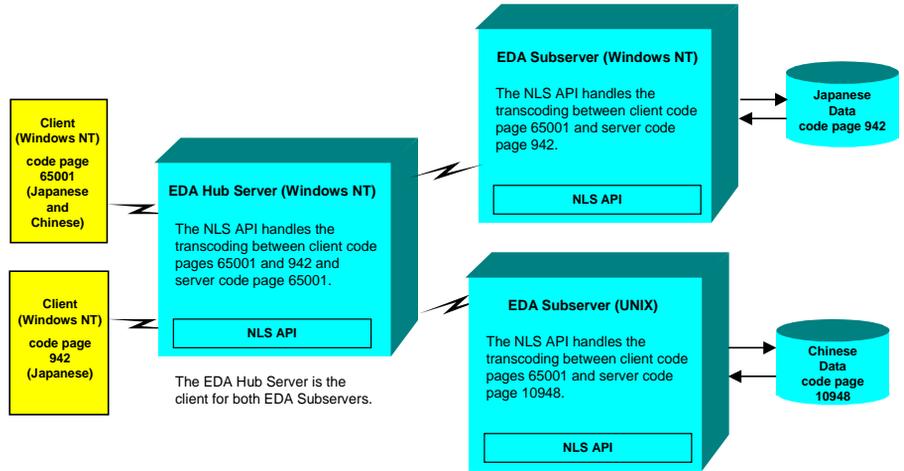
Combining Data in Different Scripts With Unicode

The following is an example of a Unicode solution to combine two unrelated scripts such as Japanese and Chinese.

1. The two data sources have a common index key with a common range of values to join them. One data source uses the Japanese Shift-JIS code page 942, and the other data source uses the traditional Chinese Big 5 code page 10948.
 - a. Shift-JIS is Japanese PC data, therefore its EDA Subserver is running on a Windows NT platform.
 - b. In this example, Big 5 is traditional Chinese UNIX data, therefore its EDA Subserver is running on a UNIX platform.
2. While the two EDA Subservers have different server code page settings, both are configured to communicate with the same client code page (UTF-8 Unicode code page 65001). The UTF-8 Unicode configuration is the only solution for combining unrelated scripts.

The EDA Hub Server is the client for both EDA Subservers and therefore the server code page setting on the EDA Hub Server (UTF-8 Unicode code page 65001) is the same as the client code page setting for the EDA Subservers (UTF-8 Unicode code page 65001).

3. The EDA Hub Server has been configured to communicate with two client code pages (UTF-8 Unicode code page 65001 and Shift-JIS code page 942).
4. One Windows NT client could be a Web browser that is set to UTF-8 Unicode, so it reads both Japanese and Chinese data from code page 65001. The other Windows NT client could be a Web browser that is set to Shift-JIS, so it reads Japanese data from code page 942. However, any Chinese data this second browser encounters will be displayed incorrectly.
5. Shift JIS (Japanese) and Big 5 (Traditional Chinese) are unrelated scripts and cannot be combined without using a Unicode solution.



CHAPTER 2

Understanding National Language Versions

Topics:

- What is a National Language Version (NLV)?
- Installing National Language Versions
- Using National Language Versions
- Local Language Online Help
- Working With an Unfamiliar Local Language
- Interpreting Local Language Error Messages
- Changing the Local Language

National Language Versions (NLVs) are software products where the user interface is displayed in a local language such as French, German, Japanese, or Spanish. Information Builders provides NLVs in selected languages for business intelligence products such as WebFOCUS.

What is a National Language Version (NLV)?

A National Language Version (NLV) is a version of a software product where the entire user interface (menus, dialog boxes, screen text, Online Help systems, error messages, and utility screens) is displayed in a particular language.

Information Builders' NLV products are positional, which means every button and menu choice is always in the same relative position in the product—only the text of the labels changes. For this reason, it is easy for anyone who is familiar with an Information Builders product to support the product in a local language.

Examples of Information Builders' NLV products are the French, German, Japanese, and Spanish editions of WebFOCUS. Information Builders' NLV products include both the translated and localized user interface. NLVs also have complete National Language Support (NLS) capability, which means you can access data for any supported language, with all national characters displayed, printed, and processed correctly. For more information on National Language Support (NLS), see Chapter 1, *Understanding National Language Support*.

National Language Version Components

At Information Builders, a National Language Version (NLV) has the following user interface components translated:

- Initial program installation
- Menus, toolbars, dialog boxes, forms
- Program utilities (assistants, wizards, editors)
- Online Help, including context-sensitive Help
- Error messages

Installing National Language Versions

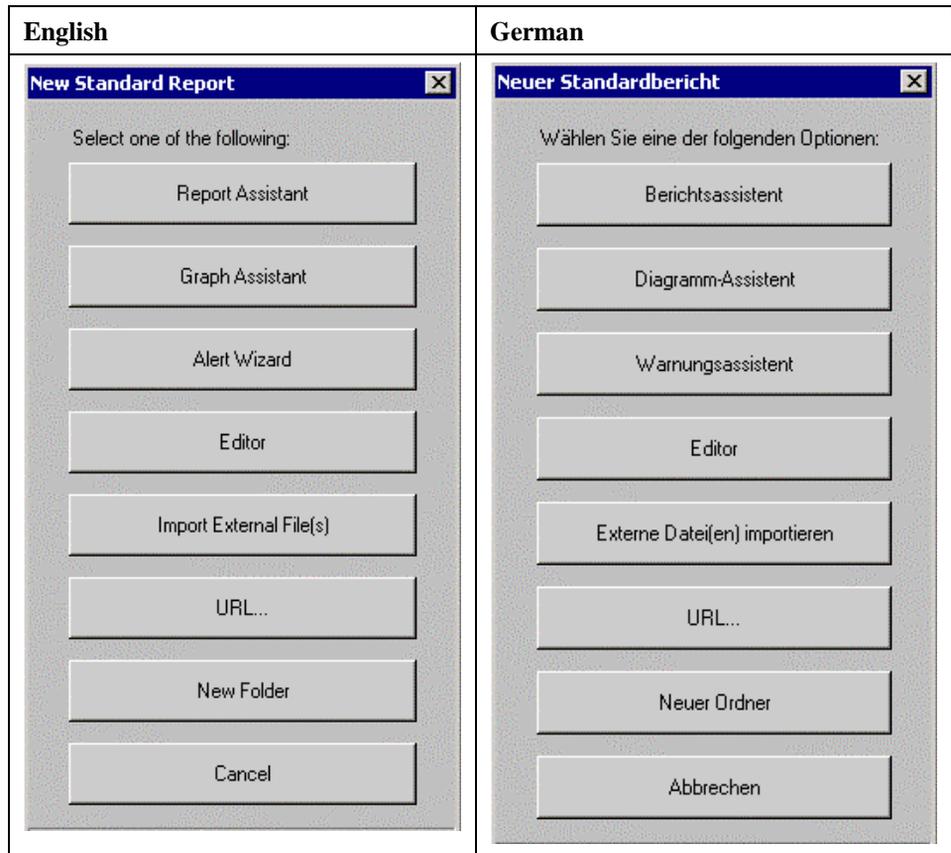
NLVs have an initial installation program in the local language (in some cases, you may be given a choice of local languages). The sequence of steps in the installation is identical to the English version of the product. For example, the following screen is from the French installation for WebFOCUS. Notice that the text is in French, but the installation has the same choices and functions as the English version, or any other local language version.



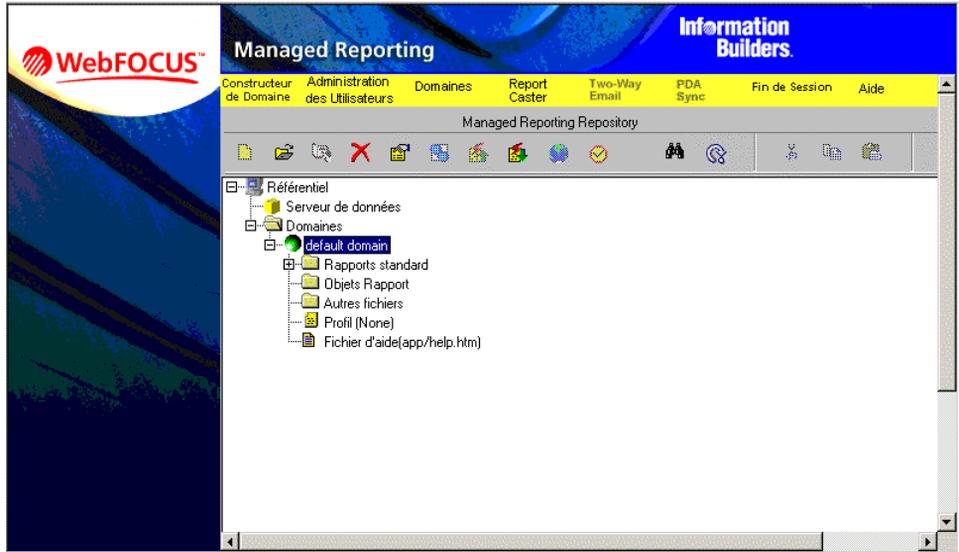
Using National Language Versions

A NLV program interface is displayed with text in the local language. The functionality of the program interface is the same for any NLV. The buttons and tabbed dialog box options are always in the same positions—only the text changes for each local language.

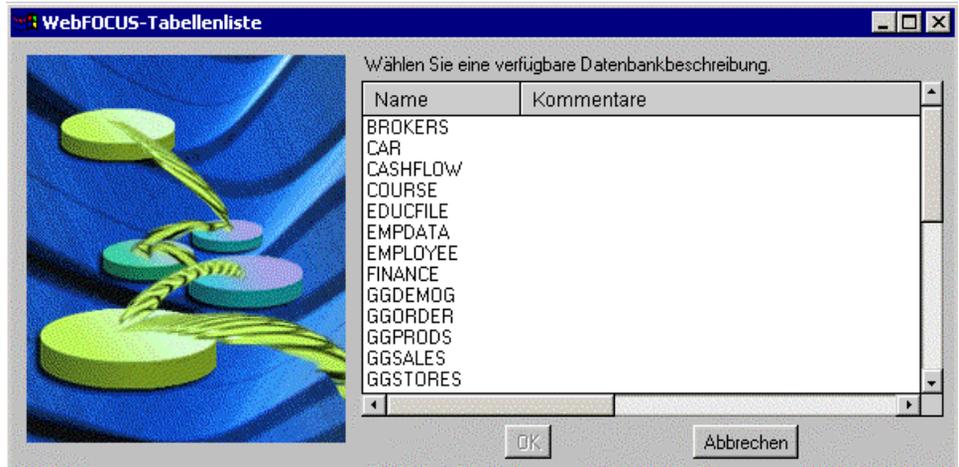
For example, the following is a side-by-side comparison of the WebFOCUS New Standard Report screen for English (left) and German (right). The options for choosing Assistants and other actions are in the same positions, but the text is displayed in the local language.



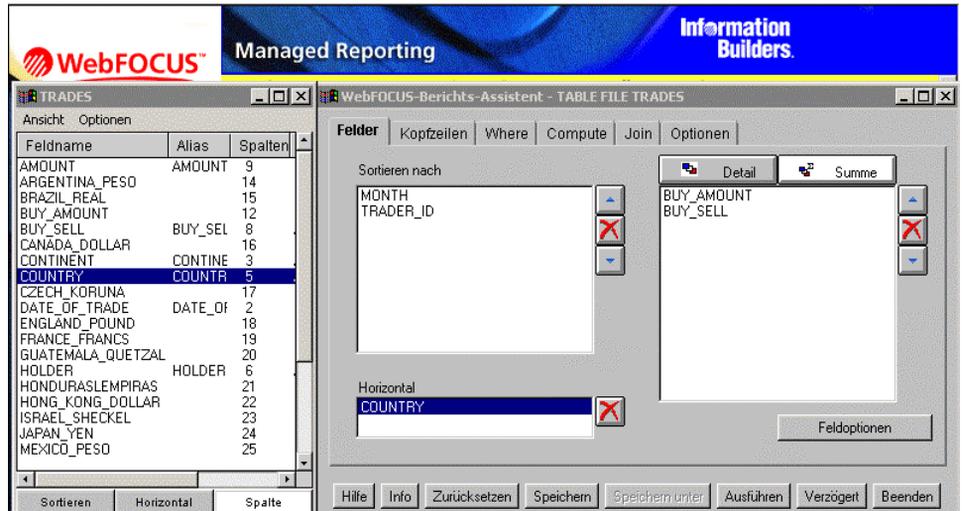
The following screen is the Managed Reporter Environment for the French WebFOCUS NLV. Notice that the names of the WebFOCUS facilities (Domain Builder, User Administrator) are translated. These facilities are translated so that you can work efficiently with the various Information Builders' NLV products. The Assistants and other utilities will also be displayed in the local language.



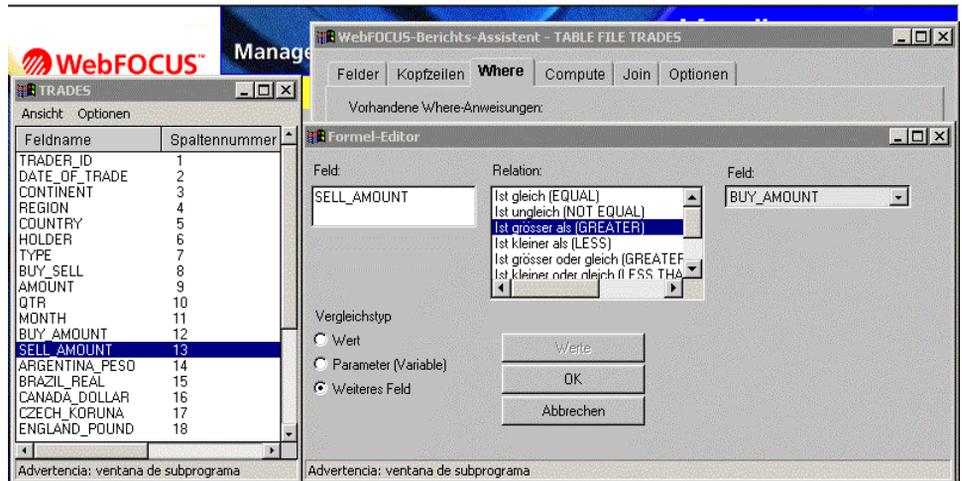
In the following screen, the file names for FOCUS data sources such as CAR and EMPLOYEE are not translated. However, they may contain text data with national characters, such as German umlauts and French accented letters.



The following screens are from the German WebFOCUS NLV. Notice in the below Report Assistant screen, the Where, Compute, and Join tabs are not translated. In general, FOCUS keywords (Where, Compute, Join) are not translated because these keywords are international terms within the Information Builders worldwide user community. A procedure (FOCEXEC) has the same syntax when created with any NLV.



In the following German Format Editor screen, FOCUS keywords are adjacent to the local language keyword to assist you in working more efficiently with the NLV product.



Important:

The Far Eastern CJK languages (Chinese, Japanese, and Korean) use a more complicated method of generating ideographic input than simply hitting a single key for each written symbol. The CJK languages generally use an Input Method Editor (IME) utility, which is typically included in the computer operating system. There are several standard input methods for each of the CJK languages and operating systems often support several input methods. See your operating system documentation for details.

Local Language Online Help

NLV Online Help systems function identically to the English version of the Online Help system and cover the same topics. However, all the text is displayed in the local language.

The following is the table of contents of the Spanish WebFOCUS NLV Online Help system.

WebFOCUS Managed Reporting Online Help - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección http://localhost/libi_html/javaassist/help/index.htm

Contenido Índice Búsqueda

- Prefacio
- Introducción a Managed Reporting
- Entorno del Administrador
- Entorno de Informaje de Dominios
- Entorno de Informaje basado en HTML
- Describiendo datos OLAP-habilitados
- Utilizando el Panel de Control OLAP
- Creando un Informe Con el Asistente de Informes
- Creando un Gráfico Con el Asistente de Gráficos
- Confirmación Diferida
- Interfaz del Usuario de ReportCaster
- Introducción a WebFOCUS Workbench
- Manteniendo Datos y Procedimientos con el Servidor de Datos
- Utilización de Dominios
- Utilización del Editor WebFOCUS
- Utilización del Componente de Otros Archivos
- Publicando Informes en el Web
- Utilización del Visor de WebFOCUS
- Formulario de Petición para Mejoramiento de Documentación

Ayuda en Línea de WebFOCUS

 **Bienvenido a Information Builders' Ayuda en Línea**

Copyright (c) 2001 Information Builders, Inc.

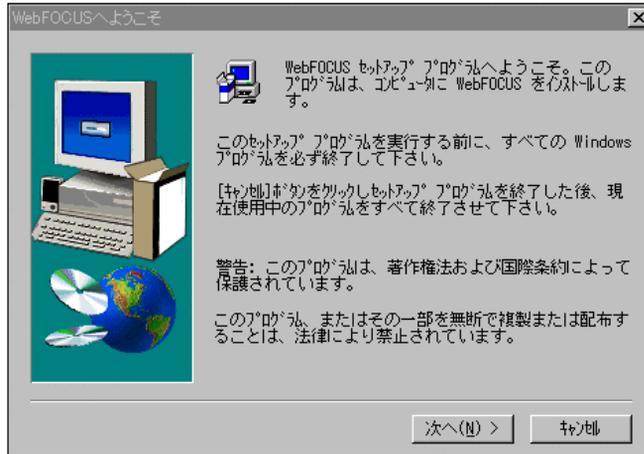
Sugerencia de Navegación

- La ficha Contenido proporciona una estructura jerarquizada de los temas más importantes.
- La ficha Índice le permite buscar por palabra clave.
- La ficha Búsqueda le permite buscar el texto del sistema entero.
- Los botones Temas Relacionados, Procedimiento, Referencia, y Ejemplo proporcionan enlaces a información adicional.
- El botón Atrás del explorador devuelve al tema previamente visto.

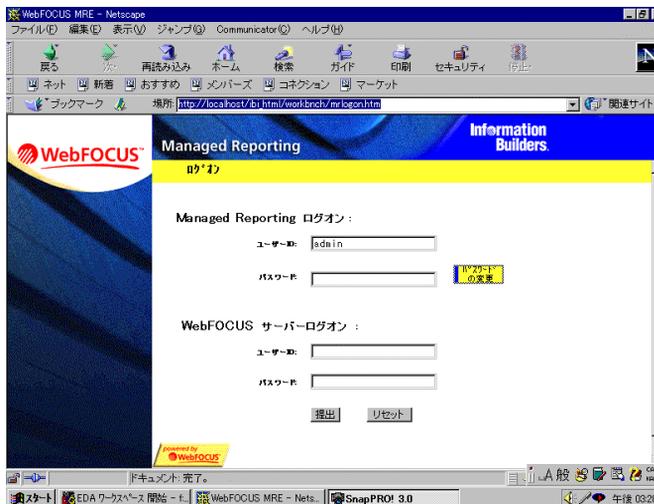
Working With an Unfamiliar Local Language

Since Information Builders' NLV products are positional, it is easy for anyone who is familiar with an Information Builders product to support an installation in a local language.

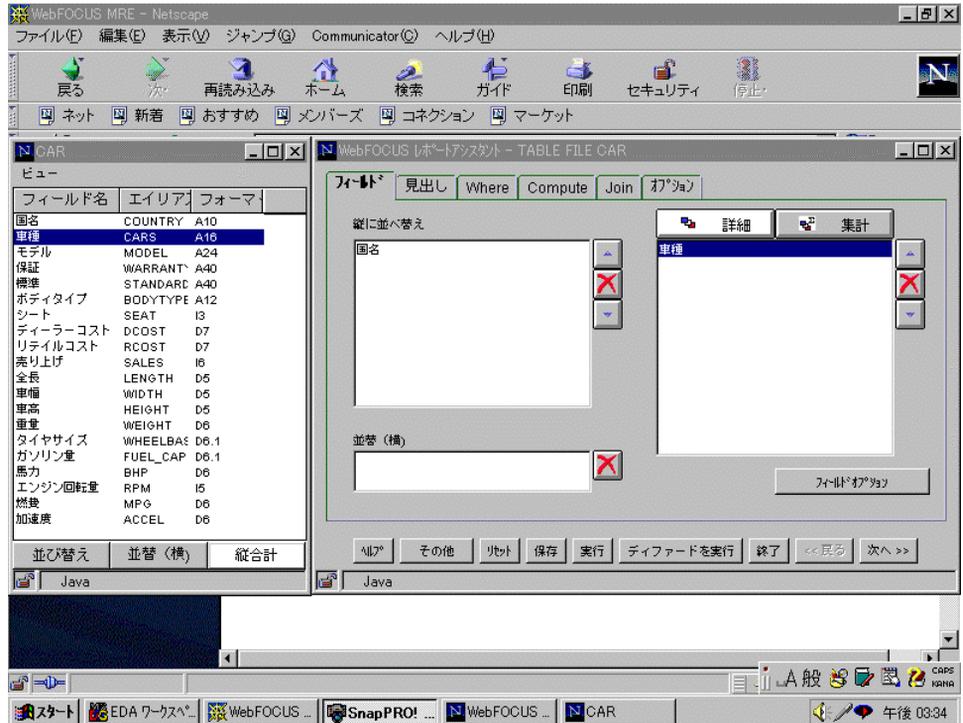
The following four screens are from the Japanese WebFOCUS NLV. The first screen is the initial installation screen and is positionally identical to the initial French WebFOCUS installation screen in *Installing National Language Versions* on page 2-3, as well as all other available WebFOCUS NLVs.



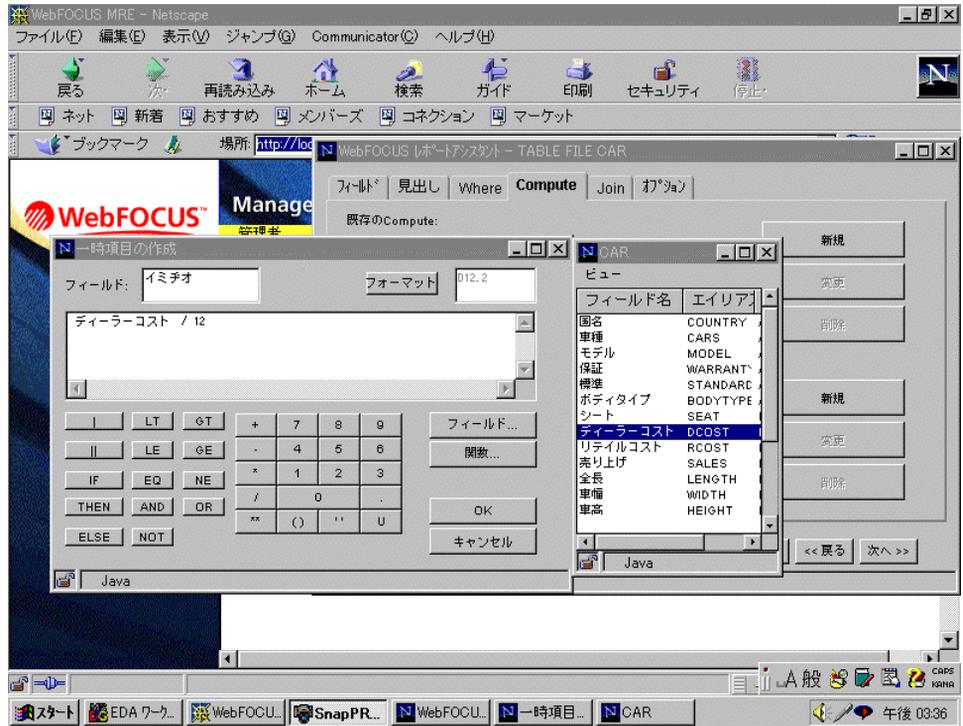
The following screen is the logon screen for the Japanese WebFOCUS Managed Reporting Environment. This screen is positionally identical to the logon screen for all available NLVs of WebFOCUS.



The following screen is the Japanese WebFOCUS Report Assistant screen and is positionally identical to the German WebFOCUS Report Assistant screen in *Using National Language Versions* on page 2-4. While the text on various buttons and tabs may be unfamiliar, the labels are identical to the labels on the German (or English) versions. The product works identically in all the various NLV products.



In the following screen, the Compute utility in WebFOCUS operates identically for the Japanese version and any other local language versions of the product.

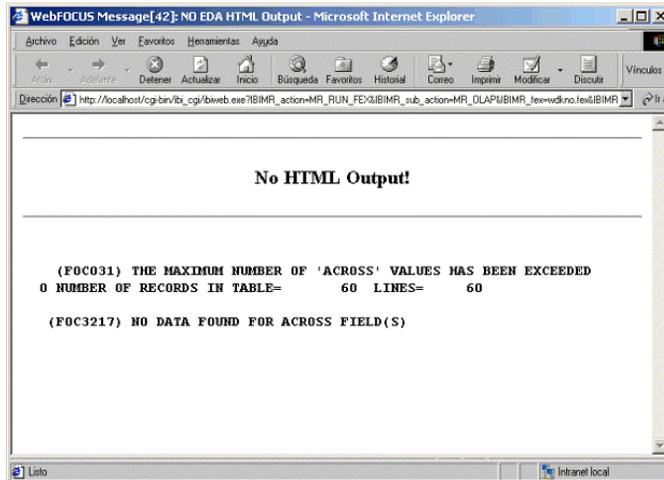


Note:

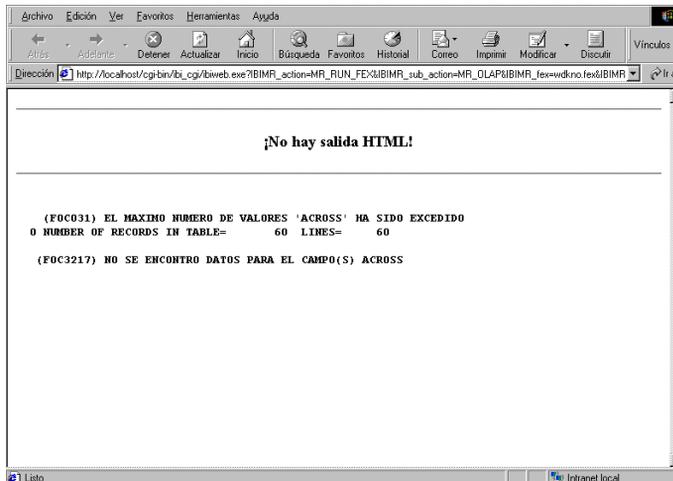
Most operating systems are now available in NLVs for many local languages. (In some cases, such as with many versions of UNIX, this is considered changing the locale setting for a single version of the operating system.) NLV operating systems work in the same manner as Information Builders' NLV products, meaning they are functionally equivalent and positional. If you are familiar with one NLV of the product, you can use any other NLV, even if you are not familiar with the local language. The major difference between the various NLVs is the provision of special Input Method Editors (IME) for Far Eastern CJK languages and other scripts.

Interpreting Local Language Error Messages

When using NLVs, product error messages display in the local language. The following screen is a numbered error message (in the form FOCnnn) from the English version of WebFOCUS. The error message screen displays a short message describing the error.



The following screen is the same error message as above, but in the Spanish NLV, generated under identical circumstances. Since the numbered error messages always refer to the same condition, if you are less familiar with the language on the screen (for example, Spanish), you can check the error message in the documentation that is in a language that you are more familiar with (for example, English) to find a more detailed description of the problem.



Changing the Local Language

In certain situations, it may be possible to change the local language of a product after installation. For example, certain versions of WebFOCUS distributed in Europe allow you to reset the language. For example, you could change the language from German to French. Contact your local Information Builders representative to determine if this possibility applies to your product.

CHAPTER 3

Automated Configuration of NLS for Windows NT and UNIX

Topics:

- Basic Initialization of Windows NT or UNIX Server NLS Services
- Basic Initialization of Windows NT or UNIX Client NLS Services

This topic describes the steps for installing and configuring the National Language Support (NLS) component for Windows NT and UNIX for both client and server. It details the basic configuration of NLS using the standardized EDA configuration tool.

Note: The NLS services installed during the basic initialization configure the client and server for standard character transcoding and are sufficient for most enterprises. Customization of your NLS configuration is only necessary if your enterprise requires support for:

- Alternate or additional code pages.
- Custom monocasing or custom sorting.

For information on customizing your NLS configuration, see Chapter 4, *Custom Configuration of NLS for Windows NT and UNIX*.

Basic Initialization of Windows NT or UNIX Server NLS Services

The basic initialization of server NLS configures the server for standard code page transcoding and is sufficient for most enterprises.

During server installation, you are asked if you would like to configure your server for NLS. If you select YES, you are prompted for all the necessary information for a basic installation of NLS. During this process, you are asked to select a server code page based on the language of the data source. Based on this server code page selection, the installation program generates the appropriate corresponding client code page settings, which are stored on the server. NLS configuration files are generated, updated, and copied into the appropriate directories. You can modify the initial NLS API settings by manually editing NLS configuration files after installation. For details on customizing your NLS configuration, see Chapter 4, *Custom Configuration of NLS for Windows NT and UNIX*.

Important:

If you anticipate the requirement of reading non-English data at any point, or if you intend to customize your NLS configuration in the future, it is recommended that you select YES when prompted to configure your server for NLS and select the default code page US English 437. During the basic installation, NLS configuration files are generated, updated, and then copied into the appropriate directories so that you can later customize the configuration. If you select NO when prompted to configure your server for NLS, the server will be setup with the default US English code page 437, but the NLS configuration files will need to be copied manually into the appropriate directories if customization is later required.

Procedure

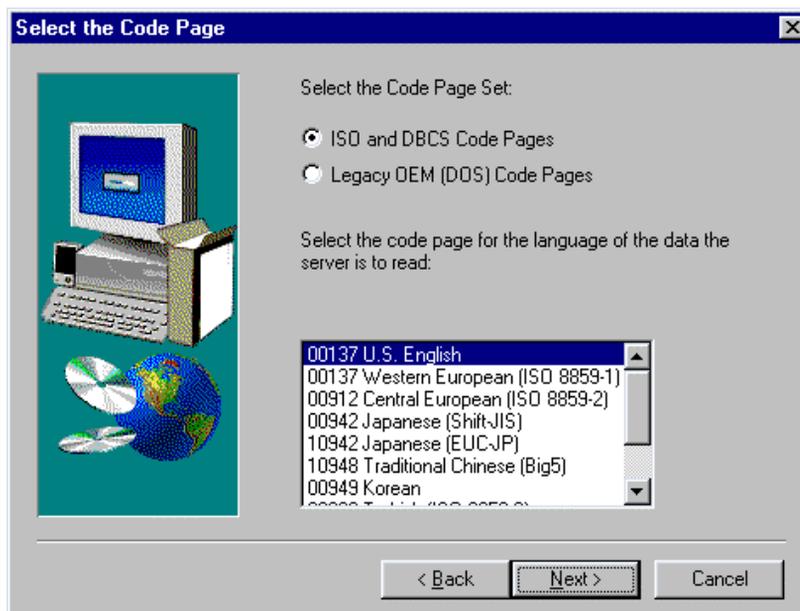
How to Configure Your Windows NT Server for Standard NLS Services

1. During Windows NT Server installation, the National Language Support window displays and prompts you to configure for National Language Support (NLS).



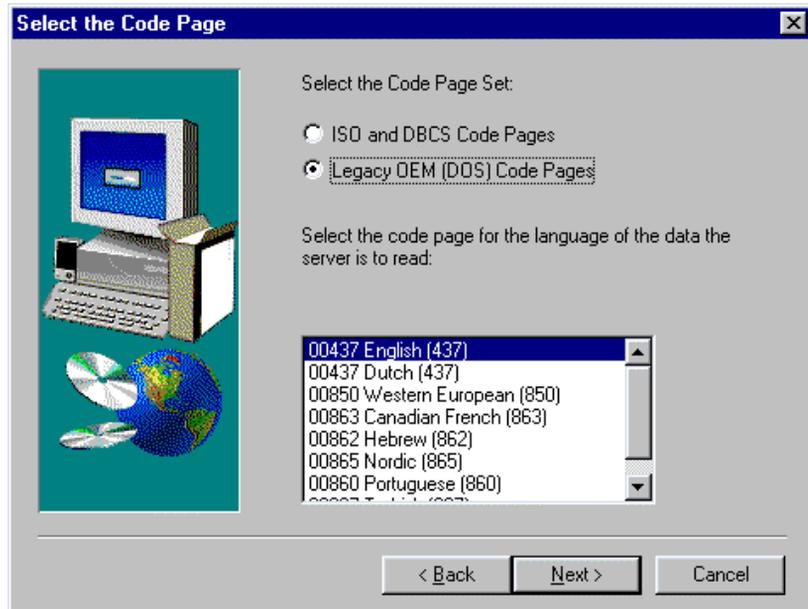
- If you choose No, proceed to step 5. Your server will be set up with the default US English code page 437.
- If you choose Yes, click Next> to continue.

2. At the Select the Code Page window, you have a choice between the following two code page sets:
 - ISO and DBCS Code Pages. This selection presents you with a list of different code pages to choose from. Within this selection of code pages, if you pick Western European (ISO 8859-1) you will get an additional screen to choose the European language of the data. This selection controls the language of the displayed error messages and other internal locale capabilities.
 - Legacy OEM (DOS) Code Pages. This selection presents you with a list of code pages used before the introduction of Windows applications. These code pages are not appropriate for current Windows technology, but they are supplied for backward compatibility purposes.



- a. If you accept the default code page set ISO and DBCS Code Pages and code page 00137 US English, click Next> and proceed to step 5.
- b. If you accept the default code page set ISO and DBCS Code Pages, and choose code page 00137 Western European (ISO 8859-1), click Next> and proceed to step 4 to select the European language of the data. This selection controls the language of the displayed error messages and other internal locale capabilities.
- c. If you accept the default code page set ISO and DBCS Code Pages, and choose a code page other than 00137 Western European (ISO 8859-1), click Next> and proceed to step 5.
- d. If you choose the code page set Legacy OEM (DOS) Code Pages, the select list will change. Proceed to step 3.

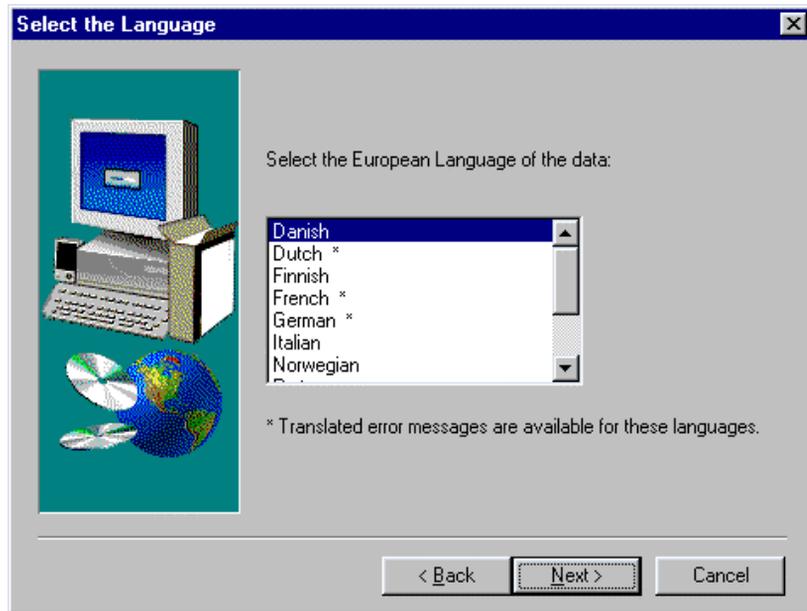
3. If you selected code page set Legacy OEM (DOS) Code Pages in step 3, the contents of the list box window changes to display available legacy code pages.



Choose a code page from the scroll list, and click Next>.

- If you choose code page 00850 Western European (850) or 00865 Nordic (865) from the list, proceed to step 4 to select the European language of the data. This selection controls the language of the displayed error messages and other internal locale capabilities.
- If you choose a code page other than 00850 Western European (850) or 00865 Nordic (865), proceed to step 5.

4. The Select the Language window displays.



Choose the European Language of the data from the scroll list, and click Next>. This selection controls the language of the displayed error messages and other internal locale capabilities.

Note: The asterisk (*) next to a language indicates that displayed error messages are translated into that language. If you choose Other, or a language without the asterisk next to it, the server error messages are displayed in English.

5. The Copying Files window displays. When the installation is finished, the Setup Complete window displays.

For details on Windows NT Server configuration, refer to the *WebFOCUS Installation and Configuration for Windows NT* manual.

Procedure**How to Configure Your UNIX Server for Standard NLS Services**

During server installation, you are prompted to configure for National Language Support (NLS). The following section covers the prompts for NLS, which assist you in configuring your UNIX Server for basic NLS services.

Prompt	Description
<pre>Configure National Language Support (NLS)? (Y/N) (Default=N):</pre>	<p>If you would like to configure for National Language Support (NLS) reply Y and follow the prompts for the remainder of this section. If you choose N, your server will be set up with the default US English code page 437.</p>
<pre>Select one of the following Code Page Sets 1 = ISO & DBCS Code Pages 2 = Legacy OEM (DOS) Code Pages (Default=1):</pre>	<p>Select Code Page Set 1 or 2.</p> <p>Note: If you choose Legacy OEM (DOS) Code Pages, you will be presented with a list of code pages used before the introduction of Windows applications. These code pages are not appropriate for current Windows technology, but they are supplied for backward compatibility purposes</p>
<pre>Select the code page for the language of the data the server is to read. ISO & DBCS Code Page Selections 1 = U.S. English [Default] 137 2 = Western European (ISO 8859-1) 137 3 = Central European (ISO 8859-2) 912 4 = Japanese (Shift-JIS) 942 5 = Japanese (EUC-JP) 10942 6 = Traditional Chinese (Big5) 10948 7 = Korean 949 8 = Turkish (ISO 8859-9) 920 9 = Unicode (UTF-8) 65001 10 = Hebrew (ISO 8859-8) 916 Legacy OEM (DOS) Code Page Selections 1 = English (437) [Default] 437</pre>	<p>There are two different lists of code pages based upon the selection of ISO and DBCS Code Pages or Legacy OEM (DOS) Code Pages.</p> <p>Note: If you select Western European from either code page set, you will receive an additional prompt to select the European language of the data. This selection controls the language of the displayed error messages and other internal locale capabilities.</p>

Prompt	Description
<pre> 2 = Dutch (437) 437 3 = Western European (850) 850 4 = Canadian French (863) 863 5 = Hebrew (862) 862 6 = Nordic (865) 865 7 = Portuguese (860) 860 8 = Turkish (867) 867 Select a number. (Default=1): </pre>	
<pre> Select the European language of the data. 1 = Danish 2 = Dutch * 3 = Finnish 4 = French * 5 = German * 6 = Italian 7 = Norwegian 8 = Portuguese 9 = Spanish * 10 = Swedish * 11 = Other * Translated error messages are available for these languages. Select a number between 1 and 11 (Default=1): </pre>	<p>Choose the European language of the data from the list. This selection controls the language of the displayed error messages and other internal locale capabilities.</p> <p>Note: The asterisk (*) next to a language indicates that displayed error messages are translated into that language. If you select 11 (Other), or a language without an asterisk next to it, the error messages will be displayed in English.</p> <p>For more information on UNIX Server configuration, refer to the <i>Installation and Configuration for UNIX</i> manual.</p>

Basic Initialization of Windows NT or UNIX Client NLS Services

The basic initialization of client NLS configures the client for standard character transcoding and is sufficient for most enterprises.

During client installation, you are asked if you would like to configure your client for NLS. If you select YES, you are prompted to select a code page based on the language of the data the client will be accepting. Based on this code page selection, the installation program generates and updates a NLS configuration file and copies it into the appropriate directory. You can modify the initial client NLS settings by manual editing the NLS configuration file after installation. For details on customizing your client NLS configuration, see Chapter 4, *Custom Configuration of NLS for Windows NT and UNIX*.

Important:

If you anticipate the requirement of reading non-English data at any point, or if you intend to customize your NLS configuration in the future, it is recommended that you select YES when prompted to configure your client for NLS and select the default code page US English 137.

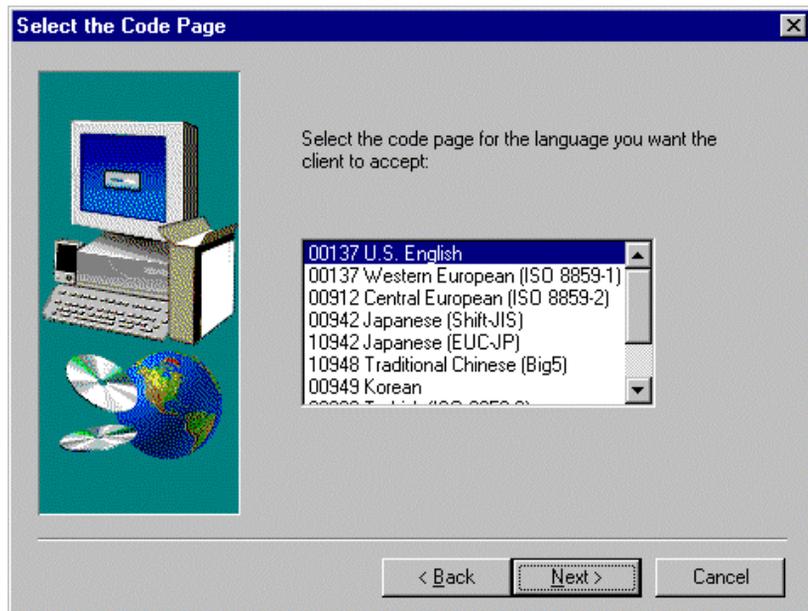
Procedure

How to Configure Your Windows NT Client for Standard NLS Services

1. During the client installation, the National Language Support window displays and prompts you to configure for National Language Support (NLS).



- If you choose No, your client will be set up with the default US English code page 137.
 - If you choose Yes, the Select the Code Page window displays.
2. Select your code page and Click Next> to accept the settings.



3. The Copying Files window displays. When the installation is finished, the Setup Complete window displays.

For more information on Windows NT client configuration, refer to the *WebFOCUS Installation and Configuration for Windows NT* manual.

Procedure**How to Configure Your UNIX Client for Standard NLS Services**

During client installation, you are prompted to configure for National Language Support (NLS). The following section covers the prompts for NLS, which assist you in configuring your UNIX client for basic NLS services.

Prompt	Description
<pre>Configure National Language Support (NLS)? (Y/N) (Default=N):</pre>	<p>If you would like to configure for National Language Support (NLS) reply Y and follow the prompts for the remainder of this section. If you choose N, your client will be set up with the default US English code page 137.</p>
<pre>Select the code page that the server provides to the client. 1 = U.S. English [Default] 137 2 = Western European (ISO 8859-1) 137 3 = Central European (ISO 8859-2) 912 4 = Japanese (Shift-JIS) 942 5 = Japanese (EUC-JP) 10942 6 = Traditional Chinese (Big5) 10948 7 = Korean 949 8 = Turkish (ISO 8859-9) 920 9 = Unicode (UTF-8) 65001 10 = Hebrew (ISO 8859-8) 916 Select a number between 1 and 10 (Default=1):</pre>	<p>Select a code page from the list.</p> <p>Note: For OpenEdition (OS390), you will be presented with a different list of code pages to choose from because Open Edition (OS390) uses the EBCDIC family of code pages rather than ASCII.</p> <p>If the desired code page is not listed, contact your local Information Builders representative for information on additional code pages.</p>
<pre>Please confirm these values with one of the following responses... Y = Accept and Start Configuration N = Start over prompting with factory defaults. Please supply confirmation:</pre>	<p>Enter Y to accept the displayed configuration information.</p> <p>For more information on UNIX client configuration, refer to the <i>Installation and Configuration for UNIX</i> manual.</p>

CHAPTER 4

Custom Configuration of NLS for Windows NT and UNIX

Topics:

- Customizing Your Windows NT or UNIX Server for NLS Services
- Changing Your Windows NT or UNIX Server Code Page Settings
- Configuring Windows NT or UNIX Server NLS Default Characteristics
- Configuring Customized Windows NT or UNIX Server NLS Monocasing
- Configuring for Customized Windows NT or UNIX Server NLS Sort Sequences
- Testing Your Windows NT or UNIX Server NLS Behavior
- Customizing Windows NT or UNIX Client NLS Services

This topic describes the steps for customizing the National Language Support (NLS) component for Windows NT and UNIX for both client and server.

Note: Customization of your NLS configuration is only necessary if your enterprise requires support for:

- Alternate or additional code pages.
- Custom monocasing or custom sorting.

The NLS services installed during the basic initialization configures the client and server for standard character transcoding and are sufficient for most enterprises.

For information of the automated configuration of NLS for Windows NT and UNIX, see Chapter 3, *Automated Configuration of NLS for Windows NT and UNIX*.

Customizing Your Windows NT or UNIX Server for NLS Services

You can customize your NLS server configuration:

- **Change the code page settings.** You can change the code page settings on the server by editing the code page generation list file (cpcodepg.nls) and then running the Transcoding Services Generation Utility (TSGU) to generate the new transcoding table. The new transcoding table is based on the list of code pages defined in the updated code page generation list file (cpcodepg.nls). For more information, see *Changing Your Windows NT or UNIX Server Code Page Settings* on page 4-4.
- **Customize NLS default characteristics.** You can change the server code page and the language the server uses for error messages by editing the NLS configuration file (nlscfg.err). For more information, see *Configuring Windows NT or UNIX Server NLS Default Characteristics* on page 4-9.
- **Customize monocasing.** You can customize the monocasing table by editing the code page definition file (cpnnnnn.nls) and then generating the new monocasing table file (casetbl.err) with the TSGU. For more information, see *Configuring Customized Windows NT or UNIX Server NLS Monocasing* on page 4-12.
- **Customize sorting.** You can customize the sorting table by editing the code page definition file (cpnnnnn.nls) and then generating the new sorting table file (sorttbl.err) with the TSGU. For more information, see *Configuring for Customized Windows NT or UNIX Server NLS Sort Sequences* on page 4-14.

Important:

Before customizing your NLS configuration, you must have already configured the server for basic NLS services. NLS configuration files are generated and updated during the basic NLS initialization and need to be in the etc subdirectory of the EDACONF directory. If you find that a certain file is not in the expected location during the NLS customization, then it may be copied from the nls subdirectory of the EDAHOME directory. For information on initializing your server for basic NLS services, see Chapter 3, *Automated Configuration of NLS for Windows NT and UNIX*.

Windows NT Users: The NLS customization steps assume you are in a DOS window.

UNIX Users: The NLS customization steps assume you are at a command line prompt.

Reference

NLS Configuration Files for Windows NT or UNIX

The following table describes the NLS configuration files:

File	Location	Description
casetbl.err	etc subdirectory of EDACONF	Monocasing table. This file contains the monocasing tables which are generated using the TSGU and are based on the code page generation list (cpcodepg.nls).
cpcodepg.nls	etc subdirectory of EDACONF	Code page generation list. This file is a list of currently active code pages. It contains one code page for the server and multiple code pages for the client.
cpnnnnn.nls	nls subdirectory of EDAHOME	Code page definition file. This file contains information on each code point value in the code page. There is a code page definition file for every code page listed in the known code page file (cpxcptbl.nls).
cpxcptbl.nls	nls subdirectory of EDAHOME	Known code page list. This file contains a list of enabled code pages for Information Builders' clients and servers.
langtbl.err	bin subdirectory of EDAHOME	Known language list file. This file contains a list of enabled language names for Information Builders' clients and servers.
nlscfg.err	etc subdirectory of EDACONF	NLS configuration file. This file controls the server code page setting and the language the server uses for error messages. There is an nlscfg.err file for both the client and server.
odin.cfg	etc subdirectory of EDACONF (of client software)	Communications configuration file. This file contains the code page setting and DBCS setting for the client.
sorttbl.err	etc subdirectory of EDACONF	Sorting table. This file contains the sorting tables which are generated using the TSGU and are based on the contents of the code page generation list (cpcodepg.nls).
trantbl.err	etc subdirectory of EDACONF	Transcoding table file. This file contains the transcoding tables which are generated using the TSGU and are based on the contents of the code page generation list (cpcodepg.nls).

Changing Your Windows NT or UNIX Server Code Page Settings

You will need to perform the following steps to change your code page configuration:

1. Identify the current code page settings.
2. Identify alternate or additional code pages by referencing the known code page file (cpxcptbl.nls).
3. Edit the code page generation list (cpcodepg.nls) file to add the code page information collected from the known code page file (cpxcptbl.nls).
4. Generate the new transcoding tables based on the updated code page generation list (cpcodepg.nls) file using the TSGU.
5. If you generated a new server code page in step 4, you will need to modify the NLS configuration file (nlscfg.err) to identify the new server code page and language used by the server for error messages.

Note: This step is only necessary if you generated a new server code page. If you generated new client code pages you do not need to modify the NLS configuration file (nlscfg.err).

Identifying Current Windows NT or UNIX Code Page Settings (Step 1 of 5)

During the basic NLS initialization, you are given a choice of server codes pages. Based on this server code page selection, the NLS API generates the corresponding client code page settings, which are stored on the server. These code page settings are reflected in the code page generation list file (cpcodepg.nls). This file is used to generate the transcoding tables with the TSGU. To identify your current code page settings, view the code page generation list file located in the etc subdirectory of the EDACONF directory.

If you choose *not* to configure your system for NLS, the server uses the default US English code page 437. The client code page setting on the server is set up to support the following default code pages:

- US Mainframe EBCDIC CP 37
- US PC ASCII CP 437
- PC Windows ANSI CP 137
- Mainframe Latin-1 1047

Example **Using the Code Page Generation List File (cpcodepg.nls) for Windows NT or UNIX**

The following is an example of the contents of a code page generation list file (cpcodepg.nls).

```
CP00037    E SBCS    US IBM MF EBCDIC code
CP00437    A SBCS    US PC ASCII code
CP00137    A SBCS    ANSI Character Set for MS-Windows
CP01047    E SBCS    IBM MF Open Systems (Latin 1)
```

Identifying Alternate or Additional Windows NT or UNIX Code Pages (Step 2 of 5)

If your server requires support for alternate or additional code pages, you can identify these code pages by referencing the known code page file (cpxcptbl.nls), which is a list of known code page files located in the nls subdirectory of the EDAHOME directory.

Note: If the desired code page is not listed in the known code page file (cpxcptbl.nls), refer to the appropriate IBM Character Data Representation Architecture (CDRA) document and create your own, or contact your local Information Builders representative for information on additional code pages.

Syntax **How to Use the Known Code Page File (cpxcptbl.nls) for Windows NT or UNIX**

```
CPnnnnn b dbcs-id description
```

where:

CP

Is the code page prefix (always CP).

nnnnn

Is the code page number.

b

Is the character type. Possible values are:

A for ASCII.

E for EBCDIC.

dbcs-id

Is the DBCS identifier (DBCSID). For possible values, see Appendix A, *TSGU and NLS Configuration File*.

description

Is a description of the code page.

Example Using the Known Code Page File (cpxcptbl.nls) for Windows NT or UNIX

The following is an example the contents of the known code page file (cpxcptbl.nls):

```
CP00037    E SBCS    US IBM MF EBCDIC code
CP00437    A SBCS    US PC ASCII code
CP00137    A SBCS    ANSI character set for MS-Windows
CP00500    E SBCS    IBM MF International European
CP00273    E SBCS    IBM MF Germany F.R./Austria
CP00277    E SBCS    IBM MF Denmark, Norway
CP00278    E SBCS    IBM MF Finland, Sweden
CP00280    E SBCS    IBM MF Italy
CP00284    E SBCS    IBM MF Spain/Latin America
CP00285    E SBCS    IBM MF United Kingdom
CP00297    E SBCS    IBM MF France
CP00424    E SBCS    IBM MF Israel(Hebrew)
CP00850    A SBCS    IBM PC Multinational
CP00856    A SBCS    IBM PC Hebrew
CP00860    A SBCS    IBM PC Portugal
CP00862    A SBCS    IBM PC Israel
CP00863    A SBCS    IBM PC Canadian French
CP00865    A SBCS    IBM PC Nordic
CP00637    A SBCS    DEC Multinational Character Set
CP00600    A SBCS    DEC German NRC Set
CP00604    A SBCS    DEC British ( United Kingdom ) NRC Set
CP00608    A SBCS    DEC Dutch ( Netherland ) NRC Set
CP00612    A SBCS    DEC Finnish ( Finland ) NRC Set
CP00616    A SBCS    DEC French ( Flemish and French/Belgian ) NRC Set
CP00620    A SBCS    DEC French Canadian NRC Set
CP00624    A SBCS    DEC Italian ( Italy ) NRC Set
CP00628    A SBCS    DEC Norwegian/Danish ( Norway,Denmark ) NRC Set
CP00632    A SBCS    DEC Portugal NRC Set
CP00636    A SBCS    DEC Spanish ( Spain ) NRC Set
CP00640    A SBCS    DEC Swedish ( Sweden ) NRC Set
```

CP00644	A	SBCS	DEC Swiss (Swiss/French and Swiss/German) NRC Set
CP00930	E	SOSI	Japanese IBM MF Katakana Code Page (cp290+cp300)
CP00939	E	SOSI	Japanese IBM MF Latin Extended (cp1027+cp300)
CP00932	A	SJIS	Japanese PC Shift-JIS (cp897+cp301)
CP00942	A	SJIS	Japanese PC Shift-JIS Extended (cp1041+cp301)
CP10942	A	EUC	Japanese PC EUC
CP10930	E	SOSIF	Japanese Mainframe (Fujitsu)
CP20930	E	SOSIH	Japanese Mainframe (Hitachi)
CP00933	E	SOSI	Korean IBM MF Extended (cp833+cp834)
CP00934	A	KPC	Korean IBM PC (cp891+cp926)
CP00944	A	KPC	Korean IBM PC Extended (cp1040+cp926)
CP00949	A	KKS	Korean KS5601 code (cp1088+cp951)
CP00935	E	SOSI	PRC IBM MF Extended (cp836+cp837)
CP00937	E	SOSI	Taiwanese IBM MF (cp37+cp835)
CP00948	A	TPC	Taiwanese IBM PC Extended (cp1043+cp927)
CP10948	A	TBIG5	Taiwanese PC BIG-5 code
CP20948	A	TNS	Taiwanese PC National Standard code
CP30948	A	TTEL	Taiwanese PC Telephone code
CP00857	A	SBCS	IBM PC Latin 5 (Turkish)
CP00920	A	IS09	ISO-8859-9 (Latin 5, Turkish)
CP01026	E	SBCS	IBM MF Turkish
CP01252	A	SBCS	ISO-8859-1 (Latin 1)
CP00819	A	SBCS	ISO-8859-1 (Latin 1, IBM Version 8X/9X undef'ed)
CP00912	A	IS02	ISO-8859-2 (Latin 2)
CP01047	E	SBCS	IBM MF Open Systems (Latin 1)
CP65001	A	UTF8	Unicode (UTF-8)
CP00916	A	IS08	ISO-8859-8 (Hebrew)

Adding Alternate or Additional Windows NT or UNIX Code Pages (Step 3 of 5)

The code page generation list file (cpcodepg.nls) is a list of code pages to generate with the TSGU. The cpcodepg.nls file is located in the etc subdirectory of the EDACONF directory and must be changed manually, using an editor, to contain the additional or alternate set of code page numbers identified in the known code page file (cpxcptbl.nls) (step 2 of 5).

Add the information for each additional or alternate code page on a separate line, one for the server and one or more for the clients accessing the server. You can copy the code page information from the known code page file (cpxcptbl.nls) into the code page generation list file (cpcodepg.nls). Note that only CP and the code page number (for example, CP00437) is required to generate the new transcoding tables. The maximum number of code page entries in the file is sixteen.

Generating Windows NT or UNIX Transcoding Tables Using the TSGU (Step 4 of 5)

The Transcoding Services Generation Utility (TSGU) program is used to generate the NLS transcoding table file (trantbl.err), based on the list of code pages defined in the code page generation list file (cpcodepg.nls). The transcoding table file (trantbl.err) contains the transcoding tables for all the possible code page combinations. You can generate a new transcoding table with the TSGU by completing the following steps:

1. Switch to the etc subdirectory of the EDACONF directory.
2. Execute edastart under the bin subdirectory of EDACONF with the TSGU parameter as follows:

For Windows NT:

```
..\bin\edastart -x TSGU
```

For UNIX:

```
../bin/edastart -x TSGU
```

3. The output of TSGU is the transcoding table file (trantbl.err), which is based on the contents of the code page generation list (cpcodepg.nls) and is located in etc subdirectory of the EDACONF directory.

Changing the Windows NT or UNIX Server Code Page Setting (Step 5 of 5)

If you generated a new server code page in step 4 and want the server to recognize this new code page, you will need to modify the NLS configuration file (nlscfg.err) to identify the new server code page and language used by the server for error messages.

The NLS configuration file is located in the etc subdirectory of the EDACONF directory. For details on changing the NLS configuration file, see *Configuring Windows NT or UNIX Server NLS Default Characteristics* on page 4-9.

Note: This step is only necessary if you generated a new server code page in step 4. If you only generated new client code pages you do not need to modify the NLS configuration file.

Configuring Windows NT or UNIX Server NLS Default Characteristics

You can change the server code page and the language the server uses for error messages by editing the NLS configuration file (nlscfg.err) located in the etc subdirectory of the EDACONF directory. You can reference the known language list file (langtbl.err), which is located in the bin subdirectory of the EDAHOME directory, for a list of enabled language names. For more information on the NLS configuration file, see Appendix A, *TSGU and NLS Configuration File*.

Error messages are only translated for German, Spanish, Swedish, French, Dutch, and Japanese. Error messages for all other languages are translated to English.

Note: Swedish and some other languages may only be partially translated.

Example

Changing the Windows NT or UNIX Server Code Page and Language

You can use an editor to change the server code page and language of the server by editing the NLS configuration file (nlscfg.err). If the known language list file (langtbl.err) contains the proper code page associated with the language, it is not necessary to specify the CODE_PAGE setting in the NLS configuration file (nlscfg.err) because changing the LANG setting sets the proper CODE_PAGE value.

Note: The LANG setting must match either the three-character language abbreviation (language ID) or the language name. The CODE_PAGE setting must match the code page number in the code page generation list file (cpcodepg.nls).

The following example illustrates how you can change English to Dutch.

```
LANG = AMENGLISH
```

```
CODE_PAGE = 137
```

to

```
LANG = DUT
```

```
CODE_PAGE = 437
```

Syntax

How to Use the Known Language List File (langtbl.err) to Identify Enabled Language Names for Windows NT or UNIX

Enabled language names are contained in the known language list file (langtbl.err), which is located in the bin subdirectory of the EDAHOME directory.

```
mmmaaaaaaaaaaaaa bbbnnnnn
```

where:

```
mmm
```

Is a language code (the international dial code).

```
aaaaaaaaaaaaa
```

Is the language name. Twelve characters is the maximum.

```
bbb
```

Is a three-character language abbreviation (language ID) that is used in the LANG setting.

```
nnnnn
```

Is the code page number.

Note: The language name and code page number in the known language list file (langtbl.err) must match the language name in the NLS configuration file (nlscfg.err) and the code page number in the code page generation list file (cpcodepg.nls).

Example

Using the Known Language List File (langtbl.err) for Windows NT or UNIX

The following is an example of the contents of the known language file (langtbl.err):

```
001AMENGLISH      00437
001ENGLISH        00437
045DANISH         DAN00850
031DUTCH          DUT00437
358FINNISH        FIN00865
033FRENCH         FRE00863
049GERMAN         GER00850
972HEBREW         HEB00862
039ITALIAN        ITA00850
081JAPANESE       JPN00942
082KOREAN         KOR00949
047NORWEGIANN     NOR00865
351PORTUGUESE    POR00860
034SPANISH        SPA00850
046SWEDIS         SWE00865
086T-CHINESE     ROC10948
090TURKISH        TUR00857
044UKENGLISH     UKE00850
```

Configuring Customized Windows NT or UNIX Server NLS Monocasing

Monocasing (also called case conversion) is the conversion of a letter from its lowercase to uppercase form (or vice versa). As part of the basic initialization, the server is set up with standard monocasing where all requests, except for data between single quotes, are converted to uppercase according to the monocasing table in the `casetbl.err` file. The monocasing table file (`casetbl.err`) converts a-z to A-Z only and is located in etc subdirectory of the EDACONF directory. If you require customized monocasing, such as special upper/lowercase accented characters, then you must modify the code page definition file (`cpnnnnn.nls`) and then generate a new NLS monocasing table file (`casetbl.err`) using the TSGU. The new monocasing table is based on the changes made to the code page definition file (`cpnnnnn.nls`).

Note: The Information Builders' user functions, `LOCASE` and `UPCASE`, respect the NLS monocasing table in the `casetbl.err` file.

Procedure

How to Customize Your NLS Monocasing Table for Windows NT or UNIX

Note: As part of the basic initialization, monocasing tables are provided for most of the common European languages. You will only need to customize the monocasing tables if you require a special monocasing configuration.

NLS monocasing involves language-sensitive (code page sensitive) uppercase and lowercase conversion. You can customize each character's attributes by completing the following steps:

1. Edit the code page definition file (`cpnnnnn.nls`) located in the etc subdirectory of the EDACONF directory. The code page definition file (`cpnnnnn.nls`) is named by the code page number and suffixed with `.nls`. For example, `cp00437.nls` contains the monocasing information for US English code page 437. For more information on the code page definition file (`cpnnnnn.nls`), see *How to Use the Code Page Definition File (cpnnnnn.nls) for Windows NT or UNIX* on page 4-16.

Note: You can reference the known code page file (`cpxcptbl.nls`) to find the name of the code page definition file.

The code page definition file (`cpnnnnn.nls`) contains the Code Point and Graphic Character Global Identifier (GCGID). Make the appropriate changes to GCGID uppercase in the third column and Character type in the fifth column. The following is a chart of a sample code page definition file layout:

Code Point	GCGID	GCGID uppercase	Sort weight	Character type
00	..NUL...	00	.
01	SS000000	01	.
02	SS010000	02	.
:				
40	SM050000	40	S
41	SS000000	41	U
42	SS000000	42	U
:				
61	LA020000	LA010000	61	L
62	LB020000	LB010000	62	L
:				
F1	ND010000	61	D
F2	ND020000	62	D
:				
FF	SP300000	FF	.

2. Edit the code page generation list (cpcodepg.nls) and add the code page definition information as described in *Changing Your Windows NT or UNIX Server Code Page Settings* on page 4-4. The updated code page generation list (cpcodepg.nls) is used to regenerate the custom monocasing table file (casetbl.err).
3. Switch to the etc directory of the EDACONF directory.
4. Execute edastart with the TSGU parameter from the etc subdirectory of the EDACONF directory as follows:

For Windows NT:

```
..\bin\edastart -x TSGU CASE
```

For UNIX:

```
../bin/edastart -x TSGU CASE
```

The TSGU generates the updated monocasing table file (casetbl.err), located in the etc subdirectory of the EDACONF directory.

Note: For additional information on modifying monocasing values in the code page definition file, refer to the IBM CDRA Library SC09-9391-00 Level 1 Registry or contact your local Information Builders representative.

Configuring for Customized Windows NT or UNIX Server NLS Sort Sequences

As part of the basic initialization, the server is set up with standard sorting where the server uses sort sequences of the binary representation of a character string according to the sorting table in the `sorttbl.err` file. The sorting table file (`sorttbl.err`) is located in the `etc` subdirectory of the `EDACONF` directory. If you require customized sorting, such as changing your sort order to account for Swedish umlauts (Ü), then you must modify the code page definition file (`cpnnnnn.nls`) and then generate a new NLS sorting table file (`sorttbl.err`) using the TSGU. The new sorting table is based on the changes made to the code page definition file (`cpnnnnn.nls`).

Procedure

How to Customize Your NLS Sort Tables for Windows NT or UNIX

Note: As part of the basic initialization, sorting tables are provided for most of the common European languages. You will only need to customize the sorting tables if you require a special sorting sequence.

If you want to use a weighted sort that accounts for characters that are out of binary sequence, you can customize the sort tables by completing the following steps.

1. Edit the code page definition file (`cpnnnnn.nls`) located in the `etc` directory of the `EDAHOME` directory. The code page definition file (`cpnnnnn.nls`) is named by the code page number and suffixed with `.nls`. For example, `CP00437.nls` contains the sorting information for US English code page 437.

Note: You can reference the known code page file (`cpxcptbl.nls`) to find the name of the code page definition file.

The code page definition file (`cpnnnnn.nls`) contains the Code Point and Graphic Character Global Identifiers (GCGID). Make the appropriate changes to the Sort weight listed in the fourth column. The following is a chart of a sample code page definition file layout:

Code Point	GCGID	GCGID uppercase	Sort weight	Character type
00	..NUL...	00	.
01	SS000000	01	.
02	SS010000	02	.
:				
40	SM050000	40	S
41	SS000000	41	U
42	SS000000	42	U
:				
61	LA020000	LA010000	61	L
62	LB020000	LB010000	62	L
:				
F1	ND010000	61	D
F2	ND020000	62	D
:				
FF	SP300000	FF	.

2. Edit the code page generation list (cpcodepg.nls) and add the code page definition information as described in *Changing Your Windows NT or UNIX Server Code Page Settings* on page 4-4. The code page generation list (cpcodepg.nls) is used to regenerate the custom sort table in the sorttbl.err file.
3. Switch to the etc directory of the EDACONF directory.
4. Execute edastart with the TSGU parameter from the etc subdirectory of the EDACONF directory as follows:

For Windows NT:

```
..\bin\edastart -x TSGU CASE
```

For UNIX:

```
../bin/edastart -x TSGU CASE
```

The TSGU generates the updated sorting table file (sorttbl.err), located in the etc subdirectory of the EDACONF directory.

Note: For additional information on modifying sorting values in the code page definition file, refer to the IBM CDRA Library SC09-9391-00 Level 1 Registry or contact your local Information Builders representative.

Syntax

How to Use the Code Page Definition File (cpnnnnn.nls) for Windows NT or UNIX

The code page definition file (cpnnnnn.nls) contains information on the characters for each code point value in the code page.

```
dd aaaaaaaaa aaaaaaaaa xx h
```

where:

dd

Is the hexadecimal code point value (00 through FF).

aaaaaaaa

Is the Graphic Character Global Identifier (GCGID).

xx

Is the Sort weight. Consists of a hexadecimal code point value (00 through FF).

h

Is the Character type. Possible values are:

L for Lower-case alphabet.

U for Upper-case alphabet.

A for Asian (non-alphabet) character.

D for Digit.

S for Special character.

C for Control (non-printable) character.

Testing Your Windows NT or UNIX Server NLS Behavior

You can test the NLS behavior and verify your configuration using the API Query tool, which is installed with the EDA Client and is part of the server install. For detailed instructions on verifying your configuration, see the:

- *EDA Server for UNIX Configuration* manual (For UNIX users).
- *EDA Server for Windows NT Installation and Configuration* manual (For Windows NT users).

Customizing Windows NT or UNIX Client NLS Services

The basic initialization of NLS configures the client for standard code page transcoding and is sufficient for most enterprises. Customization of your client NLS configuration is only necessary if your enterprise requires support for an alternate client code page. Client code page customization involves editing the NLS configuration file (nlscfg.err).

Changing Your Windows NT or UNIX Client Code Page Setting

You can change your client code page settings by manually editing the the NLS configuration file (nlscfg.err), which is located in the current application directory or in the path set by the EDANLS environment variable. For WebFOCUS, the NLS configuration file is located in the WebFOCUS client directory structure.

The client code page settings in the NLS configuration file are sufficient for all EDA Client connection. However, if you wish, you can add code page and DBCS information in the communications configuration file (odin.cfg) for documentation purposes. The communications configuration file is located in the etc subdirectory of your client EDACONF directory.

Syntax

How to Use the NLS Configuration File (nlscfg.err) for Windows NT or UNIX

You can use an editor to change the client code page setting by editing the NLS configuration file (nlscfg.err).

CODE_PAGE

Is the client code page. The hard-coded default value is 437.

DBCS

Is the DBCS ID.

Example

Changing the NLS Configuration File (nlscfg.err) for Windows NT or UNIX

You can use an editor to change the client code page and language the client uses for error messages by editing the NLS configuration file (nlscfg.err). The following is an example of a typical NLS configuration file:

```
CODE_PAGE    = 942
DBCS         = SJIS
```

Syntax

How to Use the Communications Configuration File (odin.cfg) for Windows NT or UNIX

The NLS settings for CODE_PAGE and DBCS are located in the communications block of the communications configuration file (odin.cfg).

CODE_PAGE

Is the client code page. The hard-coded default value is 437.

DBCS

Is the DBCS ID.

CHAPTER 5

Configuring NLS for MVS

Topics:

- Configuring Your MVS Server for NLS Services
- Changing Your MVS Server Code Page Settings
- Configuring MVS Server NLS Default Characteristics
- Configuring Customized MVS Server NLS Monocasing
- Configuring for Customized MVS Server NLS Sort Sequences
- Modifying the Server JCL
- Customizing MVS Client NLS Services

This topic describes the steps for configuring National Language Support (NLS) for MVS for both client and server.

Note: By default, the client and server support the US EBCDIC code page 37. Customization of your NLS configuration is only necessary if your enterprise requires support for:

- Alternate or additional code pages.
- Custom monocasing or custom sorting.

Configuring Your MVS Server for NLS Services

You can customize your NLS server configuration to:

- **Change your code page settings.** You can change the code page settings on the server by editing the code page generation list file (CPCODEPG) and then running the Transcoding Services Generation Utility (TSGU) to generate the new transcoding table. The new transcoding table is based on the list of code pages defined in the updated code page generation list file (CPCODEPG). For more information, see *Changing Your MVS Server Code Page Settings* on page 5-4.
- **Customize NLS default characteristics.** You can change the server code page setting and the language the server uses for error messages by editing the NLS configuration file (NLSCFG). For more information, see *Configuring MVS Server NLS Default Characteristics* on page 5-9.
- **Customize monocasing.** You can customize the monocasing table by editing the code page definition file (CPNNNN) and then generating the new monocasing table file (CASETBL) with the TSGU. For more information, see *Configuring Customized MVS Server NLS Monocasing* on page 5-12.
- **Customize sorting.** You can customize the sorting table by editing the code page definition file (CPNNNN) and then generating the new sorting table file (SORTTBL) with the TSGU. For more information, see *Configuring for Customized MVS Server NLS Sort Sequences* on page 5-14.

Note: Customization of your NLS configuration is only necessary if your enterprise requires support for alternate or additional code pages, or custom monocasing and sorting.

Reference NLS Configuration Files for MVS

The following table describes the NLS configuration files:

Member	Data Set	Description
CASETBL	<i>qualif</i> .EDANLS.DATA	Monocasing table. This file contains the monocasing tables which are generated using the TSGU procedure and are based on the code page generation list (CPCODEPG).
CPCODEPG	<i>qualif</i> .EDACTL.DATA	Code page generation list. This file is a list of currently active code pages. It contains one code page for the server and multiple code pages for the client.
CPNNNNN	<i>qualif</i> .EDACTL.DATA	Code page definition file. This file contains information on each code point value in the code page. There is a code page definition file for every code page listed in the known code page file (CPXCPTBL).
CPXCPTBL	<i>qualif</i> .EDACTL.DATA	Known code page list. This file contains a list of enabled code pages for Information Builders' clients and servers.
LANGTBL	<i>qualif</i> .EDAMSG.DATA	Known language list file. This file contains a list of enabled language names for Information Builders' clients and servers.
NLSCFG	<i>qualif</i> .EDAMSG.DATA	NLS configuration file. This file controls the server code page setting and the language the server uses for error messages. There is an NLSCFG file for both the client and server.
CS3CLNT	<i>qualif</i> .INSTALL.DATA	Communications configuration file. This file contains the code page setting and DBCS setting for the client.
SORTTBL	<i>qualif</i> .EDANLS.DATA	Sorting table. This file contains the sorting tables which are generated using the TSGU and are based on the contents of the code page generation list (CPCODEPG).
TRANTBL	<i>qualif</i> .EDANLS.DATA	Transcoding table file. This file contains the transcoding tables which are generated using the TSGU and are based on the contents of the code page generation list (CPCODEPG).

Changing Your MVS Server Code Page Settings

You will need to perform the following steps to change your code page configuration:

1. Identify the current code page settings.
2. Identify alternate or additional code pages by referencing the known code page file (CPXCPTBL).
3. Edit the code page generation list file (CPCODEPG) with the code page information collected from the known code page file (CPXCPTBL).
4. Generate the new transcoding tables based on the updated code page generation list file (CPCODEPG) using the TSGU.
5. If you generated a new server code page in step 4, you will need to modify the NLS configuration file (NLSCFG) to identify the new server code page and language used by the server for error messages.

Note: This step is only necessary if you generated a new server code page in step 4. If you generated new client code pages you do not need to modify the NLS configuration file (NLSCFG).

Identifying Current MVS Code Page Settings (Step 1 of 5)

Code page settings are reflected in the code page generation list file *qualif.EDACTL.DATA(CPCODEPG)*. This file contains code page settings for both client and server and it is used to generate the transcoding tables with the TSGU. To identify your current code page settings, view the code page generation list file (CPCODEPG).

During the EDA Server installation, your server is setup by default with the US EBCDIC code page 37. The client code page setting on the server is set up to support the following default code pages:

- US Mainframe EBCDIC CP 37
- US PC ASCII CP 437
- PC Windows ANSI CP 137
- Mainframe Latin-1 1047

Example Using the Code Page Generation List File (CPCODEPG) for MVS

The following is an example of a code page generation list file (CPCODEPG).

```
CP00037  E SBCS  US IBM MF EBCDIC code
CP00437  A SBCS  US PC ASCII code
CP00137  A SBCS  ANSI Character Set for MS-Windows
CP01047  E SBCS  IBM MF Open Systems (Latin 1)
```

Identifying Alternate or Additional MVS Code Pages (Step 2 of 5)

If your server requires support for alternate or additional code pages, you can identify the code pages by referencing the known code page file *qualif.EDACTL.DATA* (CPXCPTBL).

Note: If the desired code page is not listed in the known code page file (CPXCPTBL), refer to the appropriate IBM Character Data Representation Architecture (CDRA) document and create your own, or contact your local Information Builders representative for information about additional code pages.

Syntax How to Use the Known Code Page File (CPXCPTBL) for MVS

```
CPnnnnn b dbcid description
```

where:

CP

Is the code page prefix (always CP).

nnnnn

Is the code page number.

b

Is the character type. Possible values are:

A for ASCII.

E for EBCDIC.

dbcid

Is the DBCS identifier (DBCSID). For possible values, see Appendix A, *TSGU and NLS Configuration File*.

description

Is a description of the code page.

Example Using the Known Code Page File (CPXCPTBL) for MVS

The following is a listing of the contents of the known code page file (CPXCPTBL):

```
CP00037  E SBCS  US IBM MF EBCDIC code
CP00437  A SBCS  US PC ASCII code
CP00137  A SBCS  ANSI character set for MS-Windows
CP00500  E SBCS  IBM MF International European
CP00273  E SBCS  IBM MF Germany F.R./Austria
CP00277  E SBCS  IBM MF Denmark, Norway
CP00278  E SBCS  IBM MF Finland, Sweden
CP00280  E SBCS  IBM MF Italy
CP00284  E SBCS  IBM MF Spain/Latin America
CP00285  E SBCS  IBM MF United Kingdom
CP00297  E SBCS  IBM MF France
CP00424  E SBCS  IBM MF Israel(Hebrew)
CP00850  A SBCS  IBM PC Multinational
CP00856  A SBCS  IBM PC Hebrew
CP00860  A SBCS  IBM PC Portugal
CP00862  A SBCS  IBM PC Israel
CP00863  A SBCS  IBM PC Canadian French
CP00865  A SBCS  IBM PC Nordic
CP00637  A SBCS  DEC Multinational Character Set
CP00600  A SBCS  DEC German NRC Set
CP00604  A SBCS  DEC British ( United Kingdom ) NRC Set
CP00608  A SBCS  DEC Dutch ( Netherland ) NRC Set
CP00612  A SBCS  DEC Finnish ( Finland ) NRC Set
CP00616  A SBCS  DEC French ( Flemish and French/Belgian ) NRC Set
CP00620  A SBCS  DEC French Canadian NRC Set
CP00624  A SBCS  DEC Italian ( Italy ) NRC Set
CP00628  A SBCS  DEC Norwegian/Danish ( Norway,Denmark ) NRC Set
CP00632  A SBCS  DEC Portugal NRC Set
CP00636  A SBCS  DEC Spanish ( Spain ) NRC Set
CP00640  A SBCS  DEC Swedish ( Sweden ) NRC Set
CP00644  A SBCS  DEC Swiss ( Swiss/French and Swiss/German ) NRC Set
```

CP00930	E SOSI	Japanese IBM MF Katakana Code Page (cp290+cp300)
CP00939	E SOSI	Japanese IBM MF Latin Extended (cp1027+cp300)
CP00932	A SJIS	Japanese PC Shift-JIS (cp897+cp301)
CP00942	A SJIS	Japanese PC Shift-JIS Extended (cp1041+cp301)
CP10942	A EUC	Japanese PC EUC
CP10930	E SOSIF	Japanese Mainframe (Fujitsu)
CP20930	E SOSIH	Japanese Mainframe (Hitachi)
CP00933	E SOSI	Korean IBM MF Extended (cp833+cp834)
CP00934	A KPC	Korean IBM PC (cp891+cp926)
CP00944	A KPC	Korean IBM PC Extended (cp1040+cp926)
CP00949	A KKS	Korean KS5601 code (cp1088+cp951)
CP00935	E SOSI	PRC IBM MF Extended (cp836+cp837)
CP00937	E SOSI	Taiwanese IBM MF (cp37+cp835)
CP00948	A TPC	Taiwanese IBM PC Extended (cp1043+cp927)
CP10948	A TBIG5	Taiwanese PC BIG-5 code
CP20948	A TNS	Taiwanese PC National Standard code
CP30948	A TTEL	Taiwanese PC Telephone code
CP00857	A SBCS	IBM PC Latin 5 (Turkish)
CP00920	A SBCS	ISO-8859-9 (Latin 5, Turkish)
CP01026	E SBCS	IBM MF Turkish
CP01252	A SBCS	ISO-8859-1 (Latin 1)
CP00819	A SBCS	ISO-8859-1 (Latin 1, IBM Version 8X/9X undef'ed)
CP00912	A SBCS	ISO-8859-2 (Latin 2)
CP01047	E SBCS	IBM MF Open Systems (Latin 1)
CP65001	A UTF8	Unicode (UTF-8)
CP00916	A SBCS	ISO-8859-8 (Hebrew)

Adding Alternate or Additional MVS Code Pages (Step 3 of 5)

The code page generation list file *qualif.EDACTL.DATA(CPCODEPG)* is a list of code pages to generate with the TSGU and must be changed manually, using an editor, to contain the additional or alternate set of code page numbers identified in the known code page file (*CPXCPTBL*) (step 2 of 5).

Add the information for each additional or alternate code page on a separate line, one for the server and one or more for the clients accessing the server. You can copy the code page information from the known code page file (*CPXCPTBL*) into the code page generation list file (*CPCODEPG*). Note that only CP and the code page number (for example, CP00037) is required to generate the new transcoding tables. The maximum number of code page entries in the file is sixteen.

Generating MVS Transcoding Tables Using the TSGU (Step 4 of 5)

The Transcoding Services Generation Utility (TSGU) program is used to generate the NLS transcoding table file *qualif.EDANLS.DATA(TRANTBL)*, based on the list of code pages defined in the code page generation list file (*CPCODEPG*). The transcoding table file (*TRANTBL*) contains the transcoding tables for all the possible code page combinations. You can generate a new transcoding table with the TSGU by completing the following steps:

1. The sample JCL to execute TSGU is located in *qualif.EDACTL.DATA(TSGUJCL)*. Edit this JCL stream to conform to your site requirements by replacing *qualif* with the appropriate value and supplying a valid JOB card.

The output of TSGU is the transcoding table file (*TRANTBL*), which is based on the contents of the code page generation list file (*CPCODEPG*). The *TRANTBL* is generated in *qualif.EDANLS.DATA*. This data set must be allocated in the EDA Server's JCL to the ddname *ERRORS*.

2. If you get a duplicate GCGID, choose which one to discard and replace it with eight periods as follows:

.....

3. The following confirms the generated information:
 - INFO shows NLS basic information.
 - INFO CP shows a list of transcoding tables.

Changing the MVS Server Code Page Setting (Step 5 of 5)

If you generate a new server code page in step 4 and want the server to recognize this new code page, you will need to modify the NLS configuration file *qualif.EDAMSG.DATA(NLSCFG)* to identify the new server code page and language used by the server for error messages. This step is only necessary if you generated a new server code page in step 4. If you only generated new client code pages you do not need to modify the NLS configuration file (NLSCFG). For details on changing the NLS configuration file, see *Configuring MVS Server NLS Default Characteristics* on page 5-9.

Configuring MVS Server NLS Default Characteristics

You can change the server code page and the language the server uses for error messages by editing the NLS configuration file *qualif.EDAMSG.DATA(NLSCFG)*. You can reference the known language list file *qualif.EDAMSG.DATA(LANGTBL)* for a list of enabled language names. For more information on the NLS configuration file (NLSCFG), see Appendix A, *TSGU and NLS Configuration File*.

Error messages are only translated for German, Spanish, Swedish, French, Dutch, and Japanese. Error messages for all other languages are translated to English.

Note: Swedish and some other languages may only be partially translated.

Example

Changing the MVS Server Code Page and Language

You can change the server code page and language of the server by editing the LANG setting in the NLS configuration file *qualif.EDAMSG.DATA(NLSCFG)*. Changing the LANG setting sets all other parameters to valid values.

Note: The LANG setting should match the three-character language abbreviation (language ID) or the language name from the known language list file *qualif.EDAMSG.DATA(LANGTBL)*.

The following example illustrates how you can change English to Dutch.

```
LANG = AMENGLISH
```

to

```
LANG = DUT
```

Syntax

How to Use the Known Language List File (LANGTBL) to Identify Enabled Language Names for MVS

Enabled language names are contained in the known language list file *qualif.EDAMSG.DATA(LANGTBL)*.

mmm aaaaaaaaaaaa bbb nnnnn cc

where:

mmm

Is a language code (the international dial code).

aaaaaaaaaaaa

Is the language name. Twelve characters is the maximum.

bbb

Is a three-character language abbreviation (language ID) that is used in the LANG setting.

nnnnn

Is the code page number.

cc

Is the hexadecimal value of the dollar symbol.

Example

Using the Known Language List File (LANGTBL) for MVS

The following is an example of the contents of the known language list file *qualif.EDAMSG.DATA(LANGTBL)*:

```
001AMENGLISH      00037
001ENGLISH        00037
045DANISH         DAN0027767
031DUTCH          DUT00037
358FINNISH        FIN0027867
033FRENCH         FRE00297
049GERMAN         GER00273
049GERMAN         GE500500
972HEBREW         HEB00424
039ITALIAN        ITA00280
081JAPANESE       JPN00930E0
081JAPANESE       JPE00939
082KOREAN         KOR00933
047NORWEGIAN     NOR0027767
351PORTUGUESE    POR00037
034SPANISH        SPA00284
046SWEDISH        SWE0027867
086T-CHINESE     ROC00937
090TURKISH        TUR01026AD
044UKENGLISH     UKE002854A
```

Configuring Customized MVS Server NLS Monocasing

Monocasing (also called Case Conversion) is the conversion of a letter from its lowercase to uppercase form (or vice versa). As part of the basic server initialization, the server is set up with standard monocasing where all requests, except for data between single quotes, are converted to uppercase according to the monocasing table (CASETBL). The monocasing table file *qualif.EDANLS.DATA(CASETBL)* converts a-z to A-Z only. If you require customized monocasing, such as special upper/lowercase accented characters, then you must modify the code page definition file (CPNNNNN) and then generate a new NLS monocasing table file (CASETBL) using the TSGU. The new monocasing table is based on the changes made to the code page definition file (CPNNNNN).

Note: The Information Builders' user functions, LOCASE and UPCASE, respect the NLS monocasing table file (CASETBL).

Procedure

How To Customize Your NLS Monocasing Table for MVS

Note: As part of the basic initialization, monocasing tables are provided for most of the common European languages. You will only need to customize the monocasing tables if you require a special monocasing configuration.

NLS monocasing involves language-sensitive (code page sensitive) uppercase and lowercase conversion. You can customize each character's attributes by completing the following steps:

1. Edit the code page definition file (CPNNNNN), which is named by the code page number. For example, CP00037 contains the monocasing information for US English code page 37. For more information on the code page definition file (CPNNNNN), see *How to Use the Code Page Definition File (CPNNNNN)* for MVS on page 5-16.

Note: You can reference the known code page file (CPXCPTBL) to find the name of the code page definition file.

The code page definition file (CPNNNNN) contains the Code Point and Graphic Character Global Identifier (GCGID). Make the appropriate changes to GCGID uppercase in the third column and Character type in the fifth column. The following is a chart of a sample code page definition file layout:

Code Point	GCGID	GCGID uppercase	Sort weight	Character type
00	..NUL...	00	.
01	SS000000	01	.
02	SS010000	02	.
:				
40	SM050000	40	S
41	SS000000	41	U
42	SS000000	42	U
:				
61	LA020000	LA010000	61	L
62	LB020000	LB010000	62	L
:				
F1	ND010000	61	D
F2	ND020000	62	D
:				
FF	SP300000	FF	.

2. Edit the code page generation list file (CPCODEPG) and add the code page definition information as described in *Changing Your MVS Server Code Page Settings* on page 5-4. The updated code page generation list (CPCODEPG) is used to regenerate the custom monocasing table file (CASETBL).
3. Execute the TSGU with the parameter CASE. The sample JCL is located in *qualif.EDACTL.DATA(TSGUJCL)*.

The TSGU generates the updated NLS monocasing table file *qualif.EDANLS.DATA (CASETBL)*.

Note: For additional information on modifying monocasing values in the code page definition file, refer to the IBM CDRA Library SC09-9391-00 Level 1 Registry or contact your local Information Builders representative.

Configuring for Customized MVS Server NLS Sort Sequences

As part of the basic server initialization, the server is set up with standard sorting where the server uses sort sequences of the binary representation of a character string according to the sorting table *qualif.EDANLS.DATA(SORTTBL)*. If you require customized sorting, such as changing your sort order to account for Swedish umlauts (Ü), then you must modify the NLS sorting table file (SORTTBL).

Procedure

How to Customize Your NLS Sort Tables for MVS

Note: As part of the basic initialization, sorting tables are provided for most of the common European languages. You will only need to customize the sorting tables if you require a special sorting sequence.

If you want to use a weighted sort that accounts for characters that are out of binary sequence, you can customize the sort tables by completing the following steps.

1. Edit the code page definition file *qualif.EDACTL.DATA(CPNNNNN)*, which is named by the code page number. For example, CP00037 contains the sorting information for US English code page 37. For more information on the code page definition file (CPNNNNN), see *How to Use the Code Page Definition File (CPNNNNN)* for MVS on page 5-16.

Note: You can reference the known code page file (CPXCPTBL) to find the name of the code page definition file.

The code page definition file (CPNNNNN) contains the Code Point and Graphic Character Global Identifier (GCGID). Make the appropriate changes to the Sort weight listed in the fourth column. The following is a chart of a sample code page definition file layout:

Code Point	GCGID	GCGID uppercase	Sort weight	Character type
00	..NUL...	00	.
01	SS000000	01	.
02	SS010000	02	.
:				
40	SM050000	40	S
41	SS000000	41	U
42	SS000000	42	U
:				
61	LA020000	LA010000	61	L
62	LB020000	LB010000	62	L
:				
F1	ND010000	61	D
F2	ND020000	62	D
:				
FF	SP300000	FF	.

2. Edit the code page generation list file (CPCODEPG) and add the code page definition information as described in *Changing Your MVS Server Code Page Settings* on page 5-4. The updated code page generation list (CPCODEPG) is used to regenerate the custom sorting table file (SORTTBL).
3. Execute the TSGU with the parameter CASE. The sample JCL is located in *qualif.EDACTL.DATA(TSGUJCL)*.

The TSGU generates the updated NLS sorting table file *qualif.EDANLS.DATA* (SORTTBL).

Note: For additional information on modifying monocasing values in the code page definition file, refer to the IBM CDRA Library SC09-9391-00 Level 1 Registry or contact your local Information Builders representative.

Syntax

How to Use the Code Page Definition File (CPNNNNM) for MVS

The code page definition file *qualif.EDACTL.DATA(CPNNNNM)* contains information on the characters for each code point value in the code page.

```
dd aaaaaaaaa aaaaaaaaa xx h
```

where:

dd

Is the hexadecimal code point value (00 through FF).

aaaaaaaa

Is the Graphic Character Global IDentifier (GCGID).

xx

Is the Sort weight. Consists of a hexadecimal code point value (00 through FF).

h

Is the character type. Possible values are:

L for Lower-case alphabet.

U for Upper-case alphabet.

A for Asian (non-alphabet) character.

D for Digit.

S for Special character.

C for Control (non-printable) character.

Modifying the Server JCL

Concatenate *qualif.EDANLS.DATA* as the first data set of ddname **ERRORS** in the server JCL. This causes the server to use the NLS files in *qualif.EDANLS.DATA*.

Customizing MVS Client NLS Services

During the EDA Client installation, your client is set up by default with the US EBCDIC code page 37. Customization of your NLS configuration is only necessary if your enterprise requires support for an alternate client code page. Client code page customization involves editing the NLS configuration file *qualif.EDAMSG.DATA(NLSCFG)*.

Changing Your MVS Client Code Page Setting

You can change your client code page settings by manually editing the NLS configuration file *qualif.EDAMSG.DATA(NLSCFG)*. The client code page settings in the NLS configuration file are sufficient for all EDA Client connection. However, if you wish, you can add code page and DBCS information in the communications configuration file *qualif.INSTALL.DATA(CS3CLNT)* for documentation purposes.

Example

Using the NLS Configuration File (NLSCFG) for MVS

You can use an editor to change the client code page and language the client uses for error messages by editing the NLS configuration file. Changing the **LANG** setting sets all other parameters to valid values. The following is an example of a typical NLS configuration file:

```
LANG = AMENGLISH
```

Syntax

How to Use the Communications Configuration File (CS3CLNT) for MVS

The NLS settings for **CODE_PAGE** and **DBCS** are located in the communications block of the communications configuration file *qualif.INSTALL.DATA(CS3CLNT)*.

```
CODE_PAGE
```

Is the client code page. The hard-coded default value is 37.

```
DBCS
```

DBCS ID.

Example

Changing the Communications Configuration File (CS3CLNT) for MVS

The following is an example of the contents of the code page settings in a typical communications configuration file *qualif.INSTALL.DATA(CS3CLNT)*.

```
BEGIN
  PROTOCOL = TCP
  HOST = IBIMVS
  SERVICE = 1234
  CLASS = CLIENT
  CODE_PAGE = 500
END
```

CHAPTER 6

Configuring NLS for WebFOCUS (Windows Version)

- NLS Architecture
- Using the NLS Configuration Utility to Configure for Local Usage
- Configuring for Client/Server Usage
- Special Code Page Considerations

This topic describes the architecture of WebFOCUS (Windows version) and its use of multiple code pages between its internal components. This topic also details the use of the NLS Configuration Utility for installing and configuring code pages, and explains the appropriate settings to access remote data using a client/server configuration.

NLS Architecture

The WebFOCUS (Windows version) consists of two major components:

- **Data access.** Handles the selection of records from data sources and the generation of report output from those data sources. The data access component was developed in a DOS environment with legacy OEM (DOS) code pages as the default for interpreting data. However, the data access component can handle data as OEM (DOS) or ANSI (Windows) format.
- **User interface.** Known as FOCShell, contains the Windows screens, pulldown menus, and the GUI tools such as Report Painter, Report Assistant, and Graph Assistant. FOCShell displays report output in the viewer. It also displays FOCUS code in procedures and data description code in Master Files. FOCShell writes procedures and Master Files to disk when they are called up for creation or modification in its GUI tools. FOCShell was developed as a Windows product and can store files in either OEM (DOS) or ANSI (Windows) format.

ASCII-Transparency

Since the OEM (DOS) and ANSI (Windows) code pages are all ASCII-transparent, the English letters are always handled correctly. This means that the English letters, Arabic numerals 0-9, and associated punctuation symbols always have the same numeric values on all OEM (DOS) and ANSI (Windows) code pages. However, the national characters have different values on each of the code pages.

WebFOCUS (Windows version) calls the Windows operating system to convert between the two forms of data encoding. Windows-based personal computers have two different, coexisting code pages—an ANSI (Windows) code page for screen display and processing, and an OEM (DOS) code page for file I/O (Input/Output—reading and writing to text files). Examples of the Windows code page pairs are:

- English and Swedish systems use both ANSI (Windows) code page 1252 and OEM (DOS) code page 437.
- Systems using other Western European languages use the same ANSI (Windows) code page 1252 and the OEM (DOS) code page 850.

The following are detailed pictures of the Windows 1252 code page, DOS 437 code page, and DOS 850 code page. Since these three sample code pages are all ASCII-transparent, the hexadecimal value of 0x41 represent the same capital letter 'A' for each code page. However, since the hexadecimal value of B5 is a national character (letters not found in English), it represents different characters for each code page. For the Windows 1252 code page, B5 represents the micro sign, for DOS 437 code page, it represents a box drawing character, and for DOS 850 code page, it represents an accented acute letter 'A'.

Windows 1252 code page

	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
Code point 0	SP	0	@	P	`	p	€	UA	NBSP	°	À	Ð	à	ð
Code point Ox41= 'A'	1	!	1	(A)	Q	a	q	□	'	±	Á	Ñ	á	ñ
	2	"	2	B	R	b	r	,	‘	²	Â	Ò	â	ò
	3	#	3	C	S	c	s	f	“	£	³	Ã	Ó	ã
	4	\$	4	D	T	d	t	”	»	¤	´	Ä	Ô	ä
Code point B5= micro sign	5	%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å
	6	&	6	F	V	f	v	†	—	¦	¶	Æ	Ö	æ
	7	'	7	G	W	g	w	‡	—	§	·	Ç	×	ç
	8	(8	H	X	h	x	ˆ	ˆ	¨	,	È	Ø	è
	9)	9	I	Y	i	y	%b	™	©	‘	É	Ù	é
	A	*	:	J	Z	j	z	Š	š	ª	°	Ê	Ú	ê
	B	+	;	K	[k	{	<	>	«	»	Ë	Û	ë
	C	,	<	L	\	l		œ	œ	¬	¼	Ì	Ü	ì
	D	-	=	M]	m	}	UA	UA	SHY	½	Í	Ý	í
	E	.	>	N	^	n	~	Ž	ž	®	¾	Î	Þ	î
	F	/	?	O	_	o	DEL	UA	ÿ	¯	¿	Ï	ß	ï

DOS 437 code page

	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
Code point 0	SP	0	@	P	`	p	C	É	á	■	L	⊥	α	≡
Code point Ox41= 'A'	1	!	1	(A)	Q	a	q	ú	æ	í	⊥	⊥	β	±
	2	"	2	B	R	b	r	é	Æ	ó	■	⊥	⊥	Γ
	3	#	3	C	S	c	s	â	ô	ú	⊥	⊥	π	≤
	4	\$	4	D	T	d	t	ä	ö	ñ	⊥	⊥	Σ	∫
Code point B5=box drawing character	5	%	5	E	U	e	u	à	ò	Ñ	⊕	⊥	σ	∫
	6	&	6	F	V	f	v	â	û	ª	⊥	⊥	μ	+
	7	'	7	G	W	g	w	ç	ù	°	⊥	⊥	τ	≈
	8	(8	H	X	h	x	ê	ÿ	¿	⊥	⊥	Φ	°
	9)	9	I	Y	i	y	ë	Ö	—	⊥	⊥	Θ	•
	A	*	:	J	Z	j	z	è	Ü	¬	⊥	⊥	Ω	·
	B	+	;	K	[k	{	ï	¢	½	⊥	⊥	δ	√
	C	,	<	L	\	l		î	£	¼	⊥	⊥	∞	²
	D	-	=	M]	m	}	í	¥	«	⊥	⊥	∅	²
	E	.	>	N	^	n	~	Ä	Pts	«	⊥	⊥	€	■
	F	/	?	O	_	o	DEL	Ä	f	»	⊥	⊥	∅	NBSP

DOS 850 code page

	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
Code point 0x41= 'A'	0	SP	@	P	`	p	Ç	É	á	■	┘	ø	Ó	SHY
	1	!	1	Q	a	q	ü	æ	í	■	┘	Ð	ß	±
	2	"	2	B	R	b	r	é	Æ	ó	■	┘	Ê	Ô
	3	#	3	C	S	c	s	â	ô	ú		┘	Ë	Ò
	4	\$	4	D	T	d	t	ä	ö	ñ		┘	È	ó
Code point B5= accented acute letter 'A'	5	%	5	E	U	e	u	à	ò	Ñ	Ⓐ	┘	€	Ô
	6	&	6	F	V	f	v	â	û	ª	Ⓐ	┘	á	í
	7	'	7	G	W	g	w	ç	ù	º	Ⓐ	┘	Â	Î
	8	(8	H	X	h	x	ê	ÿ	¿	Ⓒ	┘	Ï	Þ
	9)	9	I	Y	i	y	ë	ÿ	®	┘	┘	Ï	Þ
	A	*	:	J	Z	j	z	è	Ü	¬		┘	Û	·
	B	+	;	K	[k	{	ï	ø	½	┘	┘	Ü	¹
	C	,	<	L	\	l		î	£	¼	┘	┘	Ý	³
	D	-	=	M]	m	}	ì	Ø	¡	┘	┘	Ý	²
	E	.	>	N	^	n	~	Ä	x	«	¥	┘	ÿ	■
	F	/	?	O	_	o	DEL	Å	f	»	┘	■	'	NBSP

Note: The euro symbol (€) is only supported with the ANSI (Windows) 1252 code page.

Data Interpretation

Since data source, procedure, and Master File content can each be encoded with either OEM (DOS) or ANSI (Windows) code pages, it is necessary to ensure that all three components interpret the data consistently. The EDA Server must be set appropriately for the code page format of the procedure and Master File passed to it, and the server must know the required answer set format.

The procedures stored by FOCShell can pass text as record selection values or lines of headings and footings to the data access component. Master Files can pass column titles or record selection criteria in ACCEPT clauses. Consider the following procedure:

```
TABLE FILE CENTHR
HEADING CENTER
"Société Française Noël en Fête"
"Rapport des Ressources Humaines"
SUM EMP_COUNT
RESIGN_COUNT
FIRE_COUNT
COMPUTE TotalPersonell/I3 = EMP_COUNT + RESIGN_COUNT + FIRE_COUNT;
WHERE LOCATION EQ 'Montréal'
END
```

This procedure contains national characters embedded in the heading as well as in the WHERE clause that controls selection of records from the CENTHR data source. If the data source is an OEM (DOS) data source, and the procedure has been saved by FOCShell in ANSI (Windows) code page format, then the text portions of the procedure must be converted to OEM (DOS) for use by the data server. This process involves editing and re-saving the procedure. If the data source and procedure are not configured for compatible code page formats, data values will not match the WHERE clause value.

Notice that both lines of the heading and the national characters in the WHERE clause contain national characters (letters not found in English). These symbols that will be displayed incorrectly if the OEM (DOS) to ANSI (Windows) conversion is not handled correctly.

Default FOCShell Code Page Type

FOCShell uses OEM (DOS) code pages by default to store procedures and Master Files. By default, the EDA Server reads data sources as OEM (DOS) data. The data access component calls the operating system to convert text in the answer set to ANSI (Windows) so that it can be displayed in FOCShell's Windows-based (ANSI) Report Viewer. Therefore, the format of text values in the OEM (DOS) stored procedures and Master Files will match the format of data in the data source.

If the data sources are stored in ANSI (Windows) format, and the EDA Server's conversion process is disabled, it is then necessary to change the FOCShell default for storing procedures and Master Files. Changing the FOCShell's default would result in the procedures and Master Files being stored in ANSI (Windows) format, which would be consistent with the data sources.

The FOCShell tools can use either OEM (DOS) or ANSI-(Windows) based input files. The input code determines which character set to use, converts it to the native character set of the tool, and, based on the setting on the AnsiOem flag, stores the results in either OEM (DOS) or ANSI (Windows) format. The determination of the character base is made on a line-by-line basis. If a line contains both character sets, FOCShell is unable to convert the characters. Therefore, procedures with mixed ANSI (Windows) and OEM (DOS) characters will display with only one character type and the other character type will not be converted.

Reference

Compatible Code Page Combinations for Procedures and Data Sources

In order for characters to display correctly, the procedure and data source must be using compatible code pages. The following table summarizes the appropriate combinations of code page formats for procedures and data sources.

	OEM (DOS) Procedure	ANSI (Windows) Procedure
OEM (DOS) Data	✓	X
ANSI (Windows) Data	X	✓

Procedure

How to Modify the Focshell.ini for ANSI (Windows) or OEM (DOS) Settings

The AnsiOem setting in the focshell.ini file controls how FOCShell stores procedures and data. To change the AnsiOem setting in the focshell.ini, perform the following steps:

1. Open the focshell.ini file in a text editor. The default location for the focshell.ini is *install_directory\desktop_version\home\bin* where *install_directory* is the root directory in which WebFOCUS (Windows version) is installed and *desktop_version* is the configuration subdirectory. For example:
2. Locate the AnsiOem= statement under the [International] topic.
3. To set the FOCShell to store the procedure in ANSI (Windows) format, set the AnsiOem setting to zero.

\ibi\desktop435\home\bin

[International]
AnsiOem=0

or

To set your FOCShell to store the procedure in OEM (DOS) format (the default), set the AnsiOem setting to one.

[International]
AnsiOem=1 (default)

Procedure How to Convert a Procedure or Master File

If you are reporting against an ANSI (Windows)-based data source but have previously created OEM (DOS)-based procedures, you may get invalid report output or “garbage” characters. In these situations, you must convert your procedure or Master File to match the data source character type. This conversion will also need to be done if you are reporting against DOS (OEM)-based data with ANSI (Windows)-based procedures or Master Files.

1. Open the procedure or Master File in a WebFOCUS (Windows version) tool such as the Text Editor and make any modification to the file. Note that you must make a modification to the file, otherwise the conversion will not take place.
2. Save the procedure or Master File. The modified file will be stored using the new setting in the focshell.ini file.

Default EDA Server Code Page Type

By default, the WebFOCUS (Windows version) data access component assumes it is reading data stored in OEM (DOS) code page format. It calls the operating system to convert the data to ANSI (Windows) code page format, when necessary, before passing it to FOCShell for display. Data must be converted when a Windows-based user tool reads it. Specifically, the EDA Server converts the data to ANSI (Windows) format when output is requested in the following formats:

- PDF
- HTML
- Excel 97 and Excel 2000
- PostScript

FOCSHELL’s native Report Viewer is a Windows-based product. It can handle either OEM (DOS) or ANSI (Windows) data because it checks the format of all data before displaying it and makes the appropriate adjustments.

The command `SET OEMTOANSI=ON` is in effect by default. To change the EDA Server’s default behavior and inhibit it from converting data to ANSI (Windows) format before sending it back FOCShell, the command `SET OEMTOANSI=OFF` must be in effect. You can issue this command from a procedure, a profile, or any other method of executing FOCUS code. It is necessary to alter the default EDA Server’s behavior if the data access component reads ANSI (Windows) data sources. In this case, the defaults must be changed to prevent the EDA Server from converting data that is already in ANSI (Windows) format. Otherwise, the conversion process would result in corrupted data. To inhibit the OEM (DOS) to ANSI (Windows) data conversion process, issue the command `SET OEMTOANSI=OFF`. Since this command governs the FOCUS environment, it is not necessary to re-start WebFOCUS (Windows version).

Note: When WebFOCUS (Windows version) is used in client/server mode against a remote EDA Server, `SET OEMTOANSI=OFF` is automatically in effect.

DOS and Windows Code Page Setting

When SET OEMTOANSI=ON is in effect, the data access component of WebFOCUS (Windows version) calls the DOS operating system to convert data in the answer set to ANSI (Windows). Likewise, when FOCShell stores a procedure or Master File in OEM (DOS) or ANSI (Windows) format, depending on the setting in the focshell.ini file, the DOS operating system is called to store the output. To perform these functions, the operating system uses the currently active OEM (DOS) and ANSI (Windows) code page settings, which are controlled by the operating system.

Reference

Standard DOS Code Page Settings

Use the table below to configure the compatible DOS code page associated with your data source. This ensures that the OEM (DOS) to ANSI (Windows) conversion is performed correctly by the operating system.

OS or Regional Setting	OEM (DOS) Code Page
English (United States)	437
Spanish (Traditional Sort)	850
German (Standard)	850
French (Belgian, Canadian, Standard)	850
Danish	865

Note: For the double-byte character sets that represent Asian languages, there is no distinction between OEM (DOS) and ANSI (Windows). For details see, *Configuring for Double-Byte Character Languages* on page 6-15.

Procedure

How to Verify Your DOS Code Page Setting

1. At the MS-DOS prompt, type chcp and press Enter.
2. The active DOS code page is displayed.

Tip:

You may use the DOS Editor to check if a procedure was saved using the correct DOS code page and characters. However, in order to view these characters correctly, you must change the DOS Editor's font to a True Type font.

Using the NLS Configuration Utility to Configure for Local Usage

The National Language Support (NLS) configuration utility is a stand-alone program that enables you to configure WebFOCUS (Windows version) for use with data sources located on the same machine. The configuration utility is a wizard that prompts you for all the necessary information to properly install and configure the NLS component for local usage.

Installation Considerations

Before installing NLS, for full compatibility with the operating environment you need to determine the settings for the following:

- **Active DOS code page.** The active DOS code page must match the code page being used by the data source. WebFOCUS (Windows version) calls the operating system to use the currently active DOS code page and transcodes data from the OEM (DOS) code page to ANSI (Windows). The active DOS code page can be determined by typing `chcp` at the DOS prompt. For details, see *DOS and Windows Code Page Setting* on page 6-8.
- **Procedure code page.** The character set in which the procedure has been stored on disk. The code page of the procedure can be determined by opening the procedure in an editor with a known ANSI (Windows) font. If all characters are displayed correctly, the character set was stored in ANSI format. If the characters are not displayed correctly, the character set was stored in OEM format.
- **Data source code page.** The data source code page must be determined by the database administrator.

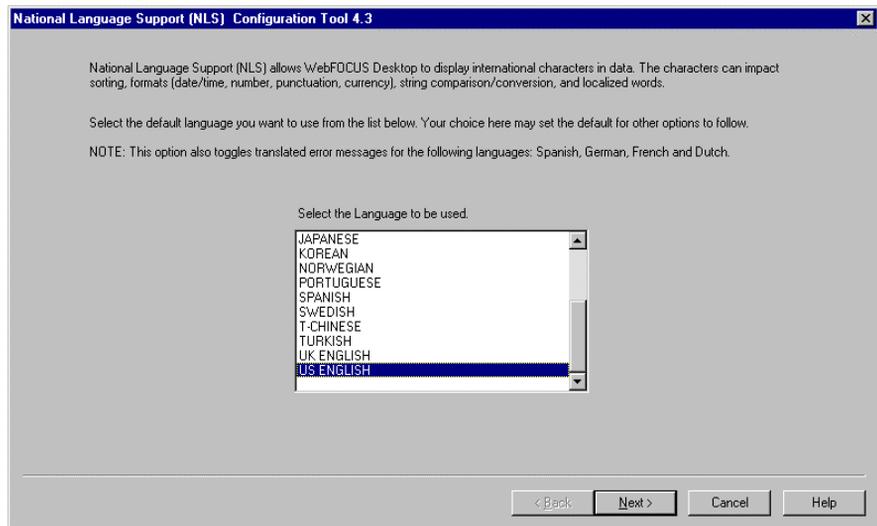
Procedure How to Configure NLS

This configuration utility should be used when WebFOCUS (Windows version) is used with local data stored in a language that contains national characters.

Note: WebFOCUS (Windows version) cannot be configured for NLS while it is running.

1. Launch the National Language Support Configuration Tool by clicking Add or Change National Language Support (NLS) from the WebFOCUS (Windows version) Utilities menu in the application start menu.

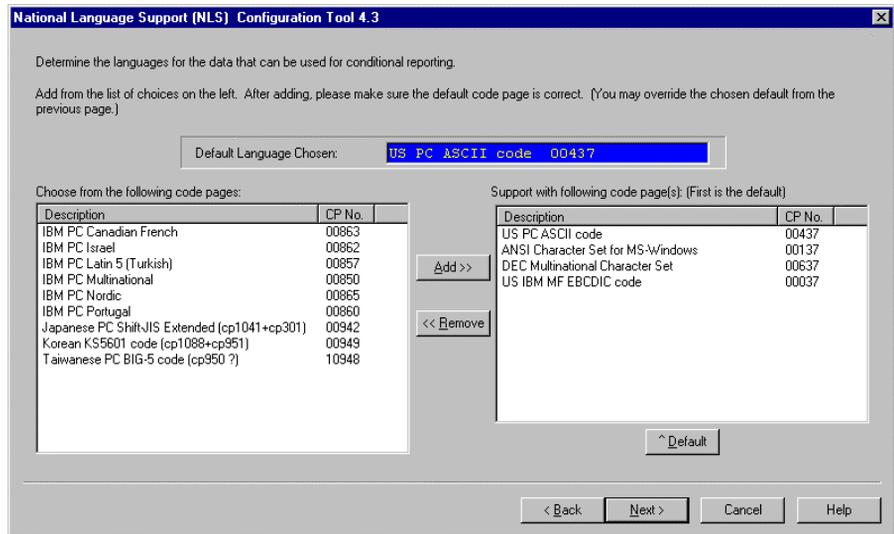
The National Language Support (NLS) Configuration Tool opens:



2. Select the language of your local PC data source from the Select the Language to be used list. This value will be reflected in the LANG= section of the NLS configuration file (nlscfg.err) located in ibi\desktop435\home\nls.

Note: Error messages are only translated for Spanish, German, French, Japanese, and Dutch. All other languages will have their error messages displayed in English.

3. Click Next to continue. The code page screen displays:



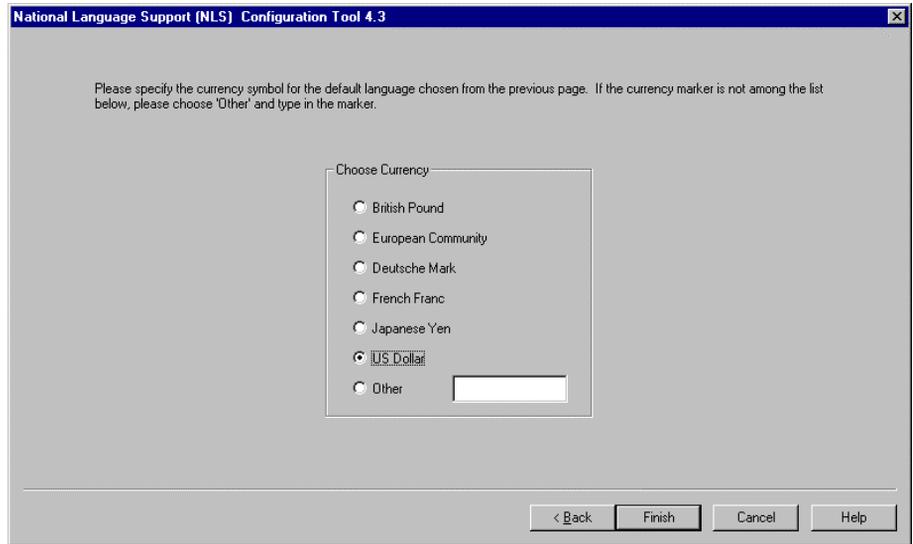
4. Add or remove code pages from the following list boxes:

Choose from the following code pages. The supported data source code pages are listed in the left window pane. If your PC data reflects more than one language, you may add multiple languages by selecting one or more code pages and clicking Add. Note that code pages for all anticipated data source languages should be selected because the configuration utility will generate and store transcoding tables for future use. The selected code pages are moved into the right window.

Support with following code page(s) (First is the Default). This list box initially contains the code pages associated with the language selected on the previous screen. Various platform code page versions for the selected language display due to cross platform portability issues with the NLS API. The first code page listed is the default. It will be active during the next execution of WebFOCUS (Windows version). Only one of the selected data source code pages can be active during a WebFOCUS (Windows version) session. To change the default code page, select a code page from the right window pane and click the Default button. The new default code page displays in the Default Language Chosen box.

To change the active data source code page, you must rerun this utility. To remove code pages from the configuration, select one or more items and click Remove.

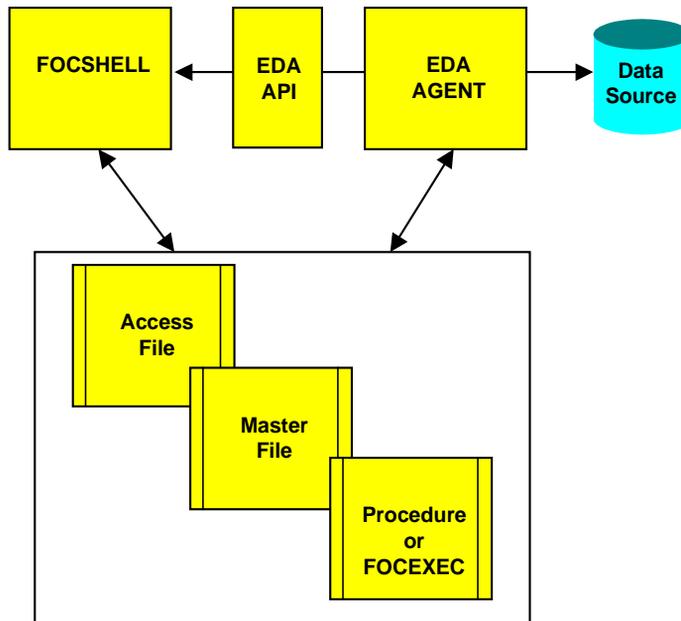
5. Click Next to continue. The currency screen displays:



6. Choose your default currency symbol. You can select a default currency to display when reporting financial information. To select a default currency symbol, click the radio button.
Note: If your desired currency symbol is not listed, select the Other radio button and then type the desired symbol in the space provided.
7. Click Finish to continue. During the final process, the Transcoding Services Generation Utility (TSGU) generates a set of transcoding tables. These binary transcoding tables contain pairs of ANSI (Windows) and OEM (DOS) code points. They control data interpretation, sorting, and upper and lower case character display.
Note: If your language selection is Japanese, or another DBCS language, there are additional manual steps that must be performed to configure NLS for WebFOCUS (Windows version). Contact your local Information Builder's representative for details.

Configuring for Client/Server Usage

You can use FOCShell as a client to a remote WebFOCUS Server or other data access engine. FOCShell sends FOCUS code to the remote EDA Server through the EDA Client. The following figure shows a sample remote access configuration:



FOCSHELL and the remote EDA Server must be configured for the same language and for consistent ANSI (Windows) and OEM (DOS) parameters. The code page setting in FOCShell's NLS configuration file (nlscfg.err) must also be supported in the server's NLS configuration file (nlscfg.err).

Note: The location of the nlscfg.err file is identified by a SETnn command in the [TscmEnv] section of focshell.ini. For example, the focshell.ini file may contain:

```
SET3=EDANLS=d:\ibi\desktop435\conf\etc
```

When WebFOCUS (Windows version) runs in local mode, the NLS configuration file (nlscfg.err) acts as the NLS configuration file for both FOCShell and the EDA Server product components. When WebFOCUS (Windows version) runs in client/server mode, this file acts as the client's NLS configuration file and the remote EDA Server has its own NLS configuration file.

Default Client/Server Configuration

The default NLS settings are appropriate for English and most European character sets. The following settings are in effect by default:

- **Server Configuration.** The server code page is set to OEM (DOS) code page 437 and the NLS configuration file (nlscfg.err) has the setting LANG=AMENGLISH. Note that for MVS, the server code page is set to OEM code page 37.
- **Client (FOCSHELL) Configuration.** FOCShell stores procedures and Master Files in OEM (DOS) format based on the setting AnsiOem=1 in its focshell.ini file.
- **Data, procedures, and Master Files.** All files accessed by client and server must be stored in OEM (DOS) code page format.

Configuring for Single-Byte Character Languages Accessing ANSI (Windows) Data

If the target data sources are ANSI (Windows) code page based, it is necessary to customize the settings of FOCShell and the EDA Server as follows:

- **Server Configuration.** The server code page is set to ANSI (Windows) code page 137 for Western European languages. For details on code page settings for other languages, see Chapter 1, *Understanding National Language Support*.

The NLS configuration file (nlscfg.err) has the setting LANG=xxx where xxx is not AMENGLISH. The server does not perform OEM (DOS) to ANSI (Windows) conversion. The conversion process is turned off automatically because of the LANG setting in the NLS configuration file (nlscfg.err).
- **Client (FOCSHELL) Configuration.** FOCShell must read and store procedures and Master Files in ANSI (Windows) format. Therefore, the setting AnsiOem=0 must be present in the focshell.ini file.
- **Data, procedures, and Master Files.** All files accessed by client and server must be stored in ANSI (Windows) code page format. If an OEM (DOS) file needs to be accessed, the CODE_PAGE parameter can be placed in the Master File but all procedures and Master Files used with this file must be converted to ANSI (Windows) format.

Configuring for Single-Byte Character Languages Accessing OEM (DOS) Data

If the target data sources are OEM (DOS) code page based, it is necessary to customize the settings of FOCShell and the EDA Server as follows:

- **Server Configuration.** The server code page is set to OEM (DOS) code page 137. The NLS configuration file (nlscfg.err) has the setting LANG=xxx where xxx is not AMENGLISH. The server does not perform OEM (DOS) to ANSI (Windows) conversion. The conversion process is turned off automatically because of the LANG setting in the NLS configuration file (nlscfg.err).
- **Client (FOCSHELL) Configuration.** FOCShell can be configured to read and store procedures and Master Files in ANSI (Windows) format (AnsiOem=0). This configuration is appropriate for the Windows output formats: Excel, Excel2k, PDF, and PS. However, national characters will not display correctly with this configuration. Alternatively, the setting AnsiOem=1 can be included in the focshell.ini file for OEM (DOS) storage of files. In this case, other output formats will display correctly, including national characters but the special Windows output types are not supported.
- **Data, procedures, and Master Files.** All files accessed by client and server must be stored in ANSI (Windows) code page format. If an OEM (DOS) file needs to be accessed, the CODE_PAGE parameter can be placed in the Master File but all procedures and Master Files used with this file must be converted to ANSI (Windows) format.

Configuring for Double-Byte Character Languages

There is no distinction between OEM (DOS) and ANSI (Windows) for the double-byte character sets that represent Asian languages. The client and server can be configured as follows:

- **Server Configuration.** The server code page is set to the appropriate code page for the target Asian language, such as Chinese, Japanese, Korean, and Taiwanese.
- **Client (FOCSHELL) Configuration.** FOCShell is configured to read and store procedures and Master Files in ANSI (Windows) format. Therefore the setting AnsiOem=0 is present in the focshell.ini file.

Special Code Page Considerations

The following items relate to using code pages and international languages:

- If the SET OEMTOANSI=OFF command is in effect but an answer set is requested in a Windows format, such as PDF, HTML, or Excel, the answer set will be produced in ANSI (Windows) format regardless of the OEMTOANSI setting.
- If the NLS configuration file (nlscfg.err) control file does not contain the statement SET LANG=AMENGLISH, then OEMTOANSI conversion is turned off. Instead of OEMTOANSI conversion, code page conversion is handled by the NLS API.

Use of Code Page Parameter in Master Files

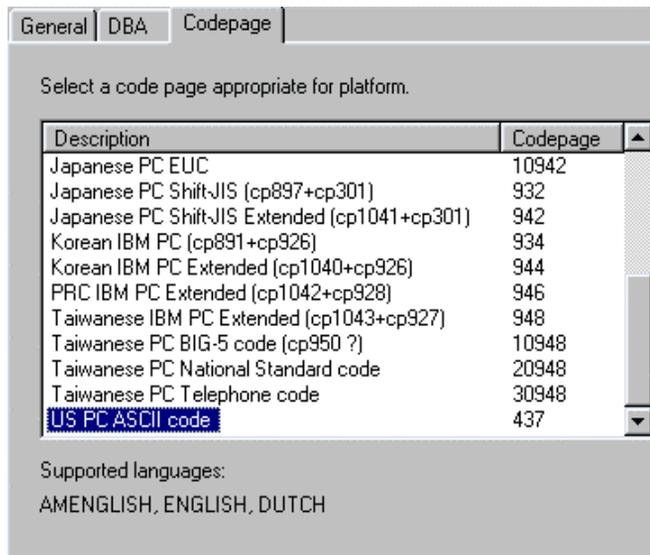
A parameter can be placed in a Master File to document the native code page of the associated data source.

Note: The code page parameter in the Master File assumes that NLS has been correctly configured for the referenced code page. It does not control interpretation of the data.

Procedure

How to Associate a Code Page to a Master File

1. In the Master Files folder, double-click on a Master File.
The Master File Editor opens.
2. Select the location path in the Master File hierarchy (left pane).
3. Select the Codepage tab (right pane). The Select a Code Page screen displays.



4. Select a code page from the list to associate to the Master File.
5. Exit the Master File Editor.
6. Click Yes when prompted to save changes to your Master File.

Displaying NLS Information: ? LANG

The ? LANG command displays the following NLS information: language, code page, dollar value, and DBCS flags.

Example

Displaying National Language Support: ? LANG

? LANG

produces a display similar to the following

```
LANGUAGE AND DBCS STATUS

Language           01/AMENGLISH  ( )
Code Page          00037
Dollar value       5B($ )
DBCS Flag          OFF(SBCS)
```

CHAPTER 7

Unicode

Topics:

- What is Unicode?
- Why Use Unicode?
- Unicode at Information Builders
- Configuring Your System for Unicode

Unicode™ is a universal character set that includes every written symbol in all the world's living languages. Unicode assigns a unique numeric value, usually expressed in hexadecimal notation, for every character. Unicode is the only way to combine data in unrelated scripts. Unlike other encoding schemes, Unicode is not dependent on operating system platform or language.

Information Builders supports UTF-8 Unicode for EDA Server version 4.3 and higher and all Java components of the WebFOCUS user-interface version 4.3.5.

What is Unicode?

Unicode is an international standard for character encoding adopted in 1992 by International Standards Organization (ISO) that includes every written symbol in all the world's living languages. The two common Unicode implementations for computer systems are UTF-8, a variable length encoding where each written symbol is represented by a one to four byte code, and UTF-16, a fixed-width encoding where each written symbol is represented by a two-byte code.

Why Use Unicode?

Unicode supports data with multiple scripts such as French, Japanese, and Hebrew, and it enables you to combine these records on a single report. Before Unicode, a computer could only process and display the written symbols of its operating system code page, which was tied to a single script. For example, if a computer could handle French, it could not also handle Japanese and Hebrew.

There is a growing trend for all new computer technologies to use Unicode for text data. In addition to Information Builders, Unicode has been adopted by industry leaders such as Microsoft, Apple, HP, IBM, Oracle, SAP, Sun, Sybase, and many others. Many of the important data sources WebFOCUS accesses now support Unicode data types. The introduction of Unicode into WebFOCUS and EDA allows you to access and work directly with Unicode data and to create UTF-8 Unicode HOLD files.

Unicode is a preferred text encoding method in browsers such as Netscape Navigator and Microsoft Internet Explorer. Unicode is also used internally in Java technologies, HTML, XML, and Windows and Office 2000. Unicode enables Information Builders' products to seamlessly handle the interface with third-party facilities that use Unicode and are integrated into Information Builders' product line.

Unicode at Information Builders

Information Builders supports UTF-8 Unicode for EDA Server version 4.3 and higher, and WebFOCUS version 4.3.5. The introduction of Unicode into Information Builders' core text-handling facilities has given Information Builders' products the flexibility to support true multilingual text and to integrate seamlessly with any future developments for all new computer technologies.

Note:

- UTF-8 Unicode is not a code page, but we have treated it as such in our product architecture. Information Builders has created a code page for UTF-8 values for all our supported scripts.
- If you are unable to read some Unicode characters in your browser, it may be because your system is not properly configured. Confirm that you are using the most recent version of your browser software and have installed the necessary fonts. For information on installing browser fonts, refer to your browser documentation.

Configuring Your System for Unicode

Information Builders' NLS API supports UTF-8 Unicode code page 65001, which is functionally identical to other supported code pages. UTF-8 Unicode is configured in the same manner as any other supported code page. For details on configuring your system for UTF-8 Unicode code page 65001, see the following topics:

- **For Windows NT and UNIX**, see Chapter 3, *Automated Configuration of NLS for NT and UNIX* and Chapter 4, *Custom Configuration of NLS for NT and UNIX*.
- **For WebFOCUS (Windows version)**, see Chapter 6, *Configuring NLS for WebFOCUS (Windows version)*.
- **For MVS**, see Chapter 5, *Configuring NLS for MVS*.

Caution:

Operating system platforms that use EBCDIC code pages (MVS and VM for mainframe, OS/400 for AS/400 midrange, and MVS Open Edition) do not support Unicode. Unicode is an ASCII/ANSI-transparent code page architecture; EBCDIC is not. Unicode does not exist for these platforms.

Therefore, you should not configure UTF-8 Unicode code page 65001 as the EDA data source code page for any EBCDIC platforms. Similarly, you should not assign EBCDIC data source code pages for ASCII platforms (Windows and UNIX).

Determining if Unicode is Necessary

When deciding whether you need to configure your system for Unicode, first determine if you need to display text in unrelated scripts. There may be situations where Unicode appears to be the only solution because you have third-party Unicode data; however, in many cases Unicode is not the only answer. For example, if you have Oracle data in a UTF-8 Unicode data type, but the text is all in Japanese and English, you do not need a full Unicode implementation. Japanese and English are not unrelated scripts and a computer system that handles Japanese text can also handle English. In this example, you have two configuration options:

- Configure the EDA Server's data source code page for UTF-8 Unicode to access the Oracle data, but set the EDA Server's client code page for a standard Japanese/English code page such as Shift-JIS for Windows NT. All the Japanese and English text data will appear correctly in a browser set to the Shift-JIS encoding.
or
- You can change the Oracle DBMS's client code page to Shift-JIS, and then configure the EDA Server for Shift-JIS on both the data source and client side.

In the second option, Oracle handles the transcoding from Unicode to Shift-JIS. Most of Information Builders' supported DBMS's have the same code page architecture as WebFOCUS and EDA, which means they have an internal component similar to Information Builders' NLS API that handles transcoding. Data source and client code page settings are typically set during installation or by editing settings in configuration files. For details, refer to the specific vendor's documentation.

Unicode is only necessary when you need to combine text in unrelated scripts, such as Japanese and Korean, or French, Japanese, and Hebrew. For example, if your UTF-8 Unicode data contains both Japanese and Korean text (unrelated scripts) that need to be displayed on a single report, UTF-8 Unicode is the only solution. In this situation, you would need to configure your entire system for UTF-8 Unicode.

Caution:

Full use of the Unicode facility may require careful attention to browser, operating system, and WebServer issues. These can include international language fonts, display, printing, and data input for unfamiliar scripts such as Japanese, Chinese, and Korean. Implementing the Unicode facility should only be undertaken if there is a real business need to combine unrelated scripts.

Defining Alphanumerics for Unicode Data Type

The UTF-8 is a variable length encoding. In general, 7-bit ASCII (familiar English letters) are one-byte, many European extended NLS characters are two-bytes, and DBCS (Japanese kanji) are three-bytes.

Familiar alphanumeric data definitions such as, SQL CHAR (n) or FOCUS An, refer to bytes, not characters. However, more complex alphanumeric data types from Oracle and other vendors may refer to characters when their client side encoding is set to UTF-8.

In Unicode, one byte is not necessarily equal to one character, so be sure to allow enough space for alphanumerics.

Understanding Your Code Page Environment

An alternative way of determining whether you need to use Unicode is to draw code page diagrams of your installation. Code page diagrams will help you identify the data source and client code page settings for each component (server, DBCS, and client) in your installation. It will also assist you in understanding where transcoding will take place. See Chapter 1, *Understanding National Language Support*, for detailed diagrams of various code page scenarios. When drawing these diagrams, you can:

- Draw blocks for each EDA Server, third-party DBCS, hub server and subserver, and client in your installation.
- Use lines to illustrate how each component is connected.

The following is an example of a scenario where Unicode is the only solution to combine data in two unrelated scripts such as Japanese and Chinese. Japanese and Chinese are unrelated scripts because, without Unicode, they will always be accessed from two different code pages on any operating system platform.

Note: For this example, the data sources may or may not be on two different operating systems.

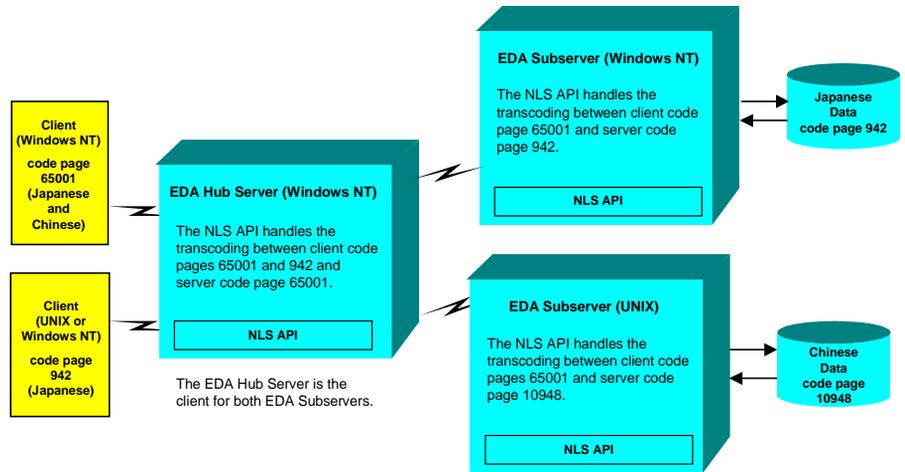
Example

Combining Data in Unrelated Scripts With Unicode

1. The two data sources must have a common index key with a common range of values to join them. One data source uses the Japanese Shift-JIS code page 942, and the other data source uses the traditional Chinese Big 5 code page 10948.
 - a. Shift-JIS is Japanese PC data, therefore its EDA Subserver is running on a Windows NT platform.
 - b. Big 5 is traditional Chinese UNIX or PC data, therefore its EDA Subserver could be running on either a UNIX or PC workstation.
2. While the two EDA Subservers have different server code page settings, both need to be configured to communicate with the same client code page 65001 (UTF-8 Unicode). A UTF-8 Unicode configuration is the only solution for joining these unrelated scripts.

The EDA Hub Server is the client for both EDA Subservers and therefore the server code page setting on the EDA Hub Server (UTF-8 Unicode code page 65001) must be the same as the client code page setting for the EDA Subservers (UTF-8 Unicode code page 65001).

3. The EDA Hub Server has been configured to communicate with two client code pages (UTF-8 Unicode code page 65001 and Shift-JIS code page 942).
4. One Windows NT client could be a Web browser that is set to UTF-8 Unicode, so it reads both Japanese and Chinese data from code page 65001. The other Windows NT client could be a Web browser that is set to Shift-JIS, so it reads Japanese data from code page 942. However, any Chinese data this second browser encounters will be displayed incorrectly.
5. Multiple EDA servers can usually run on the same computer as separate, intercommunicating processes. Therefore, in this example, (if the Big 5 data source and the two browsers are running under Windows NT), all the data sources, the three EDA servers, and the two browsers could be running on a single Windows NT computer. The processes might be running on separate computers, including a combination of UNIX and PC workstations.
6. In summary, Shift JIS (Japanese) and Big 5 (Traditional Chinese) are unrelated scripts and cannot be combined without using this type of Unicode solution.



CHAPTER 8

Troubleshooting

Topics:

- Troubleshooting Strategy
- Identifying NLS Issues
- NLS Configuration Issues
- Display Issues
- Printing Issues
- Technical Support

This topic discusses techniques for determining National Language Support (NLS) problems and identifying resolutions.

Troubleshooting Strategy

Information Builders products such as WebFOCUS and EDA are among the most advanced in the information technology marketplace. These products possess an array of sophisticated features and facilities, and can be customized to meet most user requirements.

In addition, Information Builders' products are designed to work with a wide variety of computer hardware, third-party database management systems, and data communications networks. A typical WebFOCUS or EDA installation may use multiple computers running on different operating system platforms, and may access and integrate data from a variety of different vendors' data sources. Additionally, it may use more than one data communications infrastructure to link the data sources and the computer clients and servers. These installations can be complex and an effective troubleshooting strategy can assist you in isolating the problem to a specific area of the product.

Questions to Ask

When confronted with a problem, you should start troubleshooting the situation by asking the following questions:

- Did you encounter any error messages?

Error messages should identify where, or which facility, generated the message. You may encounter error messages from the operating system, the Information Builders' product, or any third-party hardware, software, or data communications products in the installation.

Information Builders' numbered error messages (for example, FOC209) are followed by a short description of the problem. A longer description of the problem can be found in the specific product documentation.

It is important to record every error message because this information will be helpful to internal or external technical support. Once you have analyzed the error messages, you can advance to the next key question.

- Has this system configuration ever worked in its present form?

If the system worked in the past, but is not working now, you now have an important clue to understanding the problem. The next step would be to decipher what has changed with the configuration.

- Has something changed with the system configuration?

By determining what has changed with the system, you may be able to solve, or at least isolate, the problem. For example, was something added or changed in a Master File or procedure? If there are multiple steps in the application, can you determine which step caused the problem?

Try to simplify the situation by excluding third-party products. For example, if you are using an Information Builders' interface to a supported external data source such as Oracle, Sybase, Microsoft SQL Server, or DB2, can you duplicate the problem with a standard FOCUS data source, such as CAR or EMPLOYEE, or with a sequential file?

- What release of the Information Builders' product are you using?

Verify that a new version of the product was not installed since the product last functioned properly, or that the product was not upgraded with a maintenance patch or WebFOCUS Service Pack.

- Has anything changed in the computing environment?

A new piece of hardware, an upgrade to the operating system or Web browser, a different version of a third-party data source, or a change in the data communications infrastructure could be relevant to the problem you have encountered.

Observing Results

If the product worked in the past and is not working successfully now, and you have identified the settings that have changed since it last worked, you may be able to reset the product back to the initial settings. If this is possible, change one item at a time and then observe the results.

In many cases, you will not be able to set everything back to the initial settings, but by asking the general troubleshooting questions, you will have a clearer picture of the problems and the causes.

Additional Resources

The problems you encounter may have already been documented. Information Builders' products, such as WebFOCUS, have Release Notes, which is a compilation of all known problems and limitations for the specific release of your software. Additionally, the product documentation may include valuable information or useful hints about the specific feature you are trying to use.

Third-party products in your installation, including operating systems, hardware and software, and supported data sources, should have identical resources. Most products will have written documentation and a README file that provides you with the most current information on known problems.

Finally, both Information Builders and most the third-party products have World Wide Web sites with useful technical support information.

For additional information on troubleshooting and customer support, see *Information You Should Have* in the *Preface*.

Identifying NLS Issues

You may encounter NLS issues in any of the products that work together to support a WebFOCUS/EDA installation. Along with Information Builders' products, the following products may also exhibit NLS problems under certain circumstances:

- Web browsers
- Web servers
- Third-party data sources
- Data communications infrastructure
- Terminals
- Printers
- Operating systems for specific computers (the locale may be set incorrectly, or otherwise misconfigured)

An NLS problem usually manifests itself in one way—an expected national character appears as something else, such as another letter or national character, an outline box, or an unexpected symbol.

Additionally, NLS problems are typically specific. For example, text may be garbled in only one feature of a product, or on a particular line in a particular output screen. It may require a specific sequence set of steps to reliably reproduce the problem. For this reason, when you encounter NLS problems, it is important to take careful notes and then try to reproduce the problem systematically. Also, record the technical information for the installation, including versions of operating systems for component computers, versions of all software, including Web browsers, and code page settings.

A complete technical profile of the installation and a systematic reproduction of the NLS problem are the most valuable and useful information you can give to a technical support representative. This advice applies equally to the NLS problems you may encounter which are not related to the Information Builders' components of your installation.

NLS Configuration Issues

When a WebFOCUS or EDA installation has been properly configured for NLS, it should handle the supported local languages for all features and in all situations. However, configuration errors made in the initial installation can be the cause of NLS problems.

A typical NLS problem is when the national characters, either displayed on your PC or in print format, appear as something other than what is expected. For example, you may think you have properly configured your installation for Japanese, but the output may appear as a nonsensical jumble of English letters, European national characters (such as French accented letters), punctuation marks, symbols, empty boxes, solid blocks, branching lines, and other shapes.

If you suspect that you have an NLS configuration problem, the first step is to verify the client side and server side (data access) code pages. You can verify these code pages by enabling server traces and then consulting the trace file.

Note: If you have a complex EDA Hub Server with a Subserver configuration (multiple EDA Servers in a nested configuration), you will need to determine the code page settings of each server, and verify that the corresponding client and server code page settings at each Hub Server and Subserver interface match up correctly. Consult the server trace file for each server, and review the code page settings and interrelationships. See Chapter 1, *Understanding National Language Support*, for detailed examples of complex EDA Hub Server and Subserver configurations.

Procedure

How to Determine Code Pages With Server Trace for Windows NT

You can determine the client side and server side (data access) code pages for an EDA Server by consulting the server trace file.

1. Choose one of the following methods to start the server trace:
 - From the program start menu, select Diagnostic Functions and then select Workspace Start With Traces.
 - or
 - From the %EDACONF%\bin directory, type the following command in a DOS prompt:
 C:>edastart -traceon -start
2. The server trace file (tsnnnnnn.trc where nnnnnn is a sequential number starting with 000001) is saved in the %EDACONF%\edatemp directory.
3. Open the server trace file (tsnnnnnn.trc) in a text editor and search for the text string "nwnlscon". You should see three lines, which display the code page configuration of the EDA Server, similar to the following:

```

13.12.32 AG nwTraceN: CTH_CNCT_NAT_LEN is ON
13.12.32 AG nwsecon: entered
13.12.32 AG nwsecon: skip security calls - security is off.
13.12.32 AG nwsecon: OK
13.12.32 AG nwsrvcon: entered
13.12.32 AG nwnlscon: entered
13.12.32 AG nwnlscon: client code page = 137 ---> agent code page = 850
13.12.32 EC nlsdol: intlcm=00475AC0 codepg=137
13.12.32 ED nlsdol: Client Code Page: 137 dolval=24($)
13.12.32 ED nlsdol: Zapped (24) = 24 and (7C) = 7C
13.12.32 AD rtcon: enter
13.12.32 AD rtcon: executing profiles
13.12.32 AR errsta: file "EDASPROFEDAPROF *"
13.12.32 AR ezopn: request to open file "edasprof.prf" in rb mode
    
```

- The client code page is the client side code page of the EDA Server.
- The agent code page is the server (data source) code page of the EDA Server.

Procedure How to Determine Code Pages With Server Trace for UNIX

You can determine the client side and server side (data access) code pages for an EDA Server by consulting the server trace file.

1. To start the server trace, type the following command at the command line prompt from the \$EDACONF/bin directory:

```
edastart -traceon -start
```
2. The server trace file (*tsnnnnnn.trc* where *nnnnnn* is a sequential number starting with 000001) is saved in the \$EDACONF/edatemp directory.
3. Open the server trace file (*tsnnnnnn.trc*) in a text editor and search for the text string "nwnlscon". You should see three lines, which display the code page configuration of the EDA Server, similar to the following:

```

13.12.32 AG nwTraceN: CTH_CNCT_NAT_LEN is ON
13.12.32 AG nwsecon: entered
13.12.32 AG nwsecon: skip security calls - security is off.
13.12.32 AG nwsecon: OK
13.12.32 AG nwsrvcon: entered
13.12.32 AG nwnlscon: entered
13.12.32 AG nwnlscon: client code page = 137 ---> agent code page = 850
13.12.32 EC nlsdol: intlcm=00475AC0 codepg=137
13.12.32 ED nlsdol: Client Code Page: 137 dolval=24($)
13.12.32 ED nlsdol: Zapped (24) = 24 and (7C) = 7C
13.12.32 AD rtcon: enter
13.12.32 AD rtcon: executing profiles
13.12.32 AR errsta: file "EDASPROFEDAPROF *"
13.12.32 AR egobn: request to open file "edasprof.prf" in rb mode

```

- The client code page is the client side code page of the EDA Server.
- The agent code page is the server (data source) code page of the EDA Server.

Procedure How to Determine Code Pages With Server Trace for MVS

You can determine the client side and server side (data access) code pages for an EDA Server by consulting the server trace file.

1. Allocate IBITRACE in your server's start up JCL as follows:

```
//IBITRACE DD prefix.EDACTL.DATA(IBITRACE),DISP=SHR
```

2. Consult the server trace file, the jobname ddname TRnnnnnn (where nnnnnn is a sequential number starting with 000001) on your server's job.
3. Open the server trace file (TRnnnnnn.trc) in a text editor and search for the text string "nwnlscon". You should see three lines, which display the code page configuration of the EDA Server, similar to the following:

```
EDAIBM1A OUTPUT  A1  F 132  Trunc=132 Size=2805 Line=979 Col=1 Alt=0
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
secpcon: calling tssmf1
r1secver: user successfully authenticated "
r1secswt: impersonate user OSDNLS1 "
r1secswt: user successfully impersonated "
nwseccon: OK "
nwsrvcon: entered "
nwnlscon: entered "
nwnlscon: client code page = 1047 ---> agent code page = 278 "
nlsdol:  intlcm=0DE99700  codepg=1047 "
nlsdol:  Client Code Page: 1047 dolval=67(.) "
nlsdol:  Zapped (5B) = 5B and (4F) = 4F "
rtcon: enter
rtcon: executing profiles "
rtgo:    "EX4GL 'EDASPROF EDAPROF'" "
rtfex:  We've reached RTFEX "
```

- The client code page is the client side code page of the EDA Server.
- The agent code page is the server (data source) code page of the EDA Server.

Reviewing NLS Configuration Settings

Once you have determined that an error occurred in the NLS configuration, and that the client or server code page in the server trace file is not what you intended, you can correct the installation by manually changing the code page settings. For detailed information on changing the code page settings for your platform, see the following:

- For Windows NT or UNIX, Chapter 5, *Custom Configuration of NLS for Windows NT and UNIX*.
- For MVS, Chapter 5, *Configuring NLS for MVS*.

Windows NT and UNIX have automated NLS installations and configuration routines. These installation routines automate the writing of variables in the NLS configuration files and then run the Transcoding Services Generation Utility (TSGU). For more information on the automated installs, see Chapter 3, *Automated Configuration of NLS for NT and UNIX*.

Note: There is no automated install for MVS.

Code Page Consistency

When working out code page configuration issues, an important rule to follow is that the code page for the Master File and procedure (focexec) must be the same as the code page for the EDA Server. In the WebFOCUS architecture, it is possible to have two different code pages because a client could have one code page and pass a Master File or procedure to an EDA Server for execution with a different code page. In these cases, the familiar English letters will be handled correctly, but national characters may not display correctly if the client and server are using different code pages.

Display Issues

Be aware of the following issues involving display technologies and NLS:

- Web browsers typically support display of multiple encodings (code page technologies) for a variety of local languages. The settings are normally configured through options in the browser.

For example, if you are expecting Chinese output, but instead you encounter garbled text, or what looks like a random mixture of incorrect Chinese characters and other symbols, try resetting the browser to each of the supported Chinese encodings.

- Browsers usually have an “auto-detect” feature for code page encodings, which can sometimes be inaccurate or incorrect. In these cases, it is necessary to reset the browser for each encoding option for your target local language.
- Some older display technologies, such as dedicated terminals, were actually limited to certain code pages or character sets. There may have been hardware or software switches for defining national characters available for display. For example, the original IBM mainframe 327x display technology required reconfiguration of control units to display different EBCDIC code pages.

Printing Issues

The earliest computer printers used mechanisms like chains, daisy wheels, or type balls that actually printed (impressed) individual characters on paper output. For these technologies, the print mechanism must obviously include the national characters you want to type. For example, if you are using a traditional mainframe line printer with a print chain to produce reports, and you want to print the new euro symbol (€) for the European Community common currency, the print chain will have to include the € glyph.

However, today’s computer printers typically use technologies such as ink jet or laser printing that are able to print any shape, including text. Available characters for text are determined by printer firmware (fixed software routines embedded in specialized microchips), or software controlling the printer.

Be aware of the following issues involving printing technologies and NLS:

- A printer may have a set of system fonts that it uses unless instructed otherwise. These printers typically use font substitution, in which incoming text data for printing is parsed symbol by symbol, and the nearest matching symbol in a system font is used for each incoming letter. This system works for most applications, and improves performance.

However, these fonts may not support the national characters you need. If national characters do not print correctly, try using printer controls to turn off font substitution.

- To correctly print a script not normally used by the printer, you may need to specify bitmap images for text or something similar.
- Consult your printer documentation for details. Also, consult your computer's operating system and network documentation, as these options are often under operating system or network control.

Technical Support

Information Builders provides technical support for its world-wide customer base. Contact your local Information Builders' representative for detailed information on the full range of available support services.

APPENDIX A

TSGU and NLS Configuration File

Topics:

- Generating New Transcoding Tables
- The NLS Configuration File

This topic contains information about the options available with the Transcoding Services Generation Utility (TSGU) and the NLS configuration file.

Note: The directory structure and file names for the TSGU and NLS configuration file are operating system specific. Although this information is relevant to all operating systems, the Windows NT and UNIX directory structure and file names are used throughout this topic.

For information on the MVS directory structure and file names for the TSGU and NLS configuration file, see Chapter 5, *Configuring NLS for MVS*.

Generating New Transcoding Tables

The TSGU is a multi-functional utility that is used to create the transcoding table (trantbl.err), sorting table (sorttbl.err), and monocasing table (casetbl.err). You can run the TSGU manually through an interactive agent session (edastart -t), through the batch execute feature (edastart -x), or through the tsgu.bat script. For MVS, the TSGU is executed using *qualif.EDACTL.DATA(TSGUJCL)*.

Syntax

How to Use the TSGU

The syntax is

TSGU GEN

TSGU INFO *info-command*

TSGU CASE

TSGU SORT

where:

GEN

Generates the NLS transcoding table (trantbl.err). This is the default option if no parameters are supplied.

INFO

Displays information about the current NLS configuration.

CASE

Generates the custom monocasing table (casetbl.err) binary file.

SORT

Generates the custom sort sequence table (sorttbl.err) binary file.

Syntax**How to Use the TSGU Info-Commands**

The syntax is

```
TSGU INFO CP [DBCS] [nnn [nnn [...]]] [MAT [DES]]
```

```
TSGU INFO CP [SBCS] [nnn [nnn [...]]] [MAT [DES]]
```

```
TSGU INFO TRAN [idx [idx [...]]] [M] [V]
```

```
TSGU INFO CASE [nnn [nnn [...]]] [M] [V]
```

```
TSGU INFO SORT [nnn [nnn [...]]] [M] [V]
```

where:

CP

Lists transcoding tables.

DBCS

Lists only DBCS transcodings.

SBCS

Lists only SBCS transcodings.

nnn

Code page which only shows given code page transcodings (no leading zeros necessary).

idx

Index which only shows given index transcodings.

MAT

Creates a matrix report for transcoding tables.

DES

Reverses an order of code pages in a matrix report.

TRAN

Shows transcoding table contents (trantbl.err).

CASE

Shows uppercase/lowercase table contents (casetbl.err).

SORT

Shows sort table contents (sorttbl.err).

M

Masks null transcode values in tables.

V

Shows tables in vertical layout.

The NLS Configuration File

The NLS configuration file (nlscfg.err) controls the code page and the language used for error messages for both client and server and is located in the etc subdirectory of the EDACONF directory.

Syntax

How to Use the NLS Configuration File (nlscfg.err)

The following is a reference of keywords in the nlscfg.err file and their associated values:

LANG=

Is the language. The list of known languages file (langtbl.err), located in the nls subdirectory of the EDASHOME directory, contains a list of valid values. The value of LANG determines the defaults for other parameters on the server. The default value is AMENGLISH (American English).

CODE_PAGE=

Is the client or server code page. For Windows NT and UNIX, the hard-coded default value is 437. For MVS, the hard-coded default value is 37.

DBCS=

Is the DBCS identifier (DBCSID). Possible values are:

SBCS (OFF or 0) for no DBCS enabled. This is the default.

SOSI for IBM Mainframe SOSI DBCS enabled.

SJIS for Shift-JIS (Japanese).

EUC for EUC (Japanese).

KPC for IBM-PC (Korean).

KKS for KS5601 (Korean).

TPC for IBM-PC (Taiwanese).

TBIG5 for BIG-5 (Taiwanese).

TNS for National Standard (Taiwanese).

TTEL for Telephone (Taiwanese).

SOSIF for Fujitsu Mainframe (Japanese).

SOSIH for Hitachi Mainframe (Japanese).

CDN=

Is Continental Decimal Notation. Possible values are a comma (,) or a period (.). The default value is a period (.).

DATETIME=

Is the DATE/TIME format for SQL. Possible values are:

EDA for EDA type (default).

ISO for yyyy-mm-dd hh.mm.ss.

USA for mm/dd/yyyy hh:mm AM or PM.

EUR for dd.mm.yyyy hh.mm.ss.

JIS for yyyy-mm-dd hh:mm:ss.

LOCAL for any site-defined form.

The following parameters are dependent on other parameter settings:

LANG=

Overrides **CODE_PAGE** derived from **LANG** through **langtbl.err**.

CODE_PAGE=

Overrides **DBCS** derived from **CODE_PAGE** through **cpxcptbl.nls**.

DBCS=

Overrides **SHIFT_LENGTH**, **SHIFT_OUT**, **SHIFT_IN**, and **CODE_PAGE**.

If **langtbl.err** has the proper code pages, it is not necessary to specify the following parameters:

CODE_PAGE

DBCS

APPENDIX B

History of Code Pages

Topics:

- Origins of Code Pages
- Far Eastern Encodings
- Unicode
- Guidelines to Selecting a Code Page

This topic discusses the history of code pages and the various encoding strategies developed for Far Eastern languages. It also describes the implementation of Unicode as a code page architecture and presents guidelines to assist you in your code page selection.

Origins of Code Pages

Today's computer code page architectures are the direct descendants of earlier text processing technologies. Specifically, binary and numerical codes for individual letters have been used since the earliest days of the telegraph and the teletype. Punched card technology, which was the immediate predecessor of today's electronic computers, also used these binary and numerical codes.

In the 1950's, the American Standards Association (ASA) recognized the need for standardized text encoding for computing. Between 1963 and 1968, ASA introduced the American Standard Code for Information Interchange (ASCII), a 7-bit (128 value) encoding scheme that covered 32 control codes and 96 written symbols for the English language. This encoding scheme was widely implemented in data communications and computer operations as 8-bit (single-byte) codes, with the eighth bit used as a parity check for reliability.

The 7-bit ASCII, however, could not handle the additional national characters, such as French accented letters or German umlauts, used in European languages. A number of standards organizations, including ISO (International Organization for Standardization), proposed ways to include these additional letters. These proposals led to the creation of a set of 8-bit extended ASCII layouts, where the first 128 values are the same as the original English ASCII, and the higher 128 values are used either for European national characters, or for letters in an unrelated script such as Greek, Cyrillic (Russian), Hebrew, Arabic, or Thai. One important member of the 8-bit set is called Latin-1, or ANSI (American National Standards Institute). The higher 128 values include all of the non-English national characters used in Western Europe, North America, and South America.

During the decades when these encoding schemes were developed, they were simultaneously implemented as code pages to handle the related problem of text processing within electronic computers. At the time, computing was divided between IBM and "everybody else."

Beginning in 1964, IBM implemented a set of 57 proprietary Extended Binary Coded Decimal Interchange Code (EBCDIC) code pages based on its own earlier text technologies. These single-byte layouts were completely different from ASCII, but often included most of the same symbols. "Everybody else"—the other computer vendors—used ASCII, with differences in specific vendors' ASCII implementations.

Today we think of code pages as either being EBCDIC, ASCII, or ANSI—with ASCII and ANSI being closely related because they have the same values for the English letters. However, each these code page sets are actually a family of related code pages that handle languages and scripts other than English.

Far Eastern Encodings

The Far Eastern (CJK) scripts, which include Chinese, Japanese, and Korean, contain thousands of written symbols, far more than could be incorporated in 8-bit (single-byte) code pages. For this reason, an encoding solution had to be developed that could handle the CJK scripts.

The general encoding solution for the CJK scripts was developed in Japan, and it uses a 16-bit (two byte) code page for each of these scripts. Double-byte code pages can handle up to 16,536 written symbols, more than enough for most text processing applications. Different computer implementations were later developed to improve performance with double-byte codes. Today, three general strategies are used for CJK encodings.

The first strategy, modal encoding, uses a two-stage process. The first stage is mode switching (for example, between a sequence of single-byte Latin letters and double-byte Japanese kanji characters), which is signaled by an escape sequence of specific codes. The second stage is the handling of the actual bytes that represent the characters. Modal encoding methods typically use 7-bit codes. An example is Japanese JIS encoding.

Non-modal encoding, the second strategy, uses the numeric value of a text stream's bytes to decide when to switch between one and two-byte-per character modes. For example, in both the Japanese Shift-JIS and traditional Chinese Big Five encodings, a value in the range 0x20-0x7F represents the corresponding 7-bit ASCII (English) letters. However, a value in the range 0x80-0xFF indicates that the value is the first byte in a double-byte ideograph. Non-modal encoding methods typically use 8-bit codes.

The third strategy, fixed-width encoding, uses the same number of bytes to represent all of the characters available in a character set. There is no switching between one and two-byte-per-character modes. This encoding method simplifies searching, indexing, and sorting of text, but can use up a large amount of space. UTF-16 Unicode is an example of fixed-width encoding methods.

The method in which these strategies have been applied to actual code page implementation tends to be operating system specific. Some systems use both a single-byte code page (for Latin text and/or kana) and a double-byte code page (for ideographs), and switch between them. In addition, there are multiple code page solutions on the same platform for each of these languages, particularly Chinese.

Unicode

During the last decade, the Unicode Consortium, an industry standards group, and ISO have jointly developed Unicode. Unicode, sometimes referred to as ISO/IEC 10646, is a double-byte (16,536 values) encoding scheme that includes every written symbol in every living language, including Chinese, Japanese, and Korean.

Unicode is an encoding scheme, not a code page; however, it is being widely implemented as a code page architecture in innovative information technology applications. For example, many database vendors, including WebFOCUS and EDA, now provide Unicode data types. Many new computer and Web technologies use Unicode for text data, including Java technologies, HTML 4.x, the XML family of languages, and Microsoft Windows and Office 2000. Employing Unicode for text data in new computer technologies is a growing trend. Specifically, the operating system components of palm devices, organizers, Web-enabled cell phones, and set top boxes often use Unicode for all text data.

Guidelines to Selecting a Code Page

Today, with the many available languages, scripts, operating systems, and vendors, the number of code pages has greatly increased. There are now several hundred code pages in use. The following guidelines will assist you in making your code page selection.

First, code pages are organized in families and are platform-specific. There are separate families of code pages for each operating system.

The major code page families for Information Builders' products are:

- Microsoft Windows code pages.
- UNIX code pages. These code pages include the ISO 8859-x set, which is similar to the Microsoft Windows code pages.
- IBM EBCDIC code pages. These code pages are used for IBM mainframe, MVS Open Edition, and AS/400.
- DOS ASCII code pages. These code pages are intended for the original, pre-Windows IBM compatible PC DOS operating system. They are increasingly obsolete, but are used as the default code page in some EDA server configurations.

The underlying code page of your computer operating environment is determined by the language in which the operating system is generated. This code page is compatible with your operating system and its locale (language/country) setting.

Finally, the ANSI code page is extremely useful. It is the operating system code page for all Windows and UNIX systems in Western Europe, North America, and South America. Information Builders has developed code page 137, which is functionally equivalent to Microsoft Windows 1252 and UNIX ISO 8859-1 code pages. Code page 137 handles all major North American, South American, and Western European languages (except Greek) for Windows NT and UNIX.

APPENDIX C

Euro Currency and NLS SET Commands

Topics:

- Integrating the Euro Currency
- Converting Currencies
- Preparing for Currency Conversions
- Activating the Currency Data Source
- Querying the Currency Data Source in Effect
- Processing Currency Data
- Punctuating Large Numbers
- Accessing Local Language Error Messages

This topic describes the NLS SET commands for euro currency, punctuating large numbers, and local language error messages. In addition, it describes how to create and use a currency data source to convert to and from the new euro currency

Integrating the Euro Currency

With the introduction of the euro currency, businesses need to maintain books in two currencies, add new fields to their data source designs, and perform new types of currency conversions. You can perform currency conversions according to the rules specified by the European Union. To do that, you must:

- Create a currency data source with the currency IDs and exchange rates you will use. See *Creating the Currency Data Source* on page C-4 .
- Identify fields in your data sources that represent currency data. See *Identifying Fields That Contain Currency Data* on page C-8.
- Activate your currency data source. See *Activating the Currency Data Source* on page C-10.

After you complete these preliminary steps, you can perform currency conversions. See *Processing Currency Data* on page C-12 .

Operating system vendors are in the process of integrating the euro currency symbol into their environments. As the euro symbol becomes available, Information Builders will support it.

Converting Currencies

Although the euro was introduced in 11 countries of the European Union on January 1, 1999, it will not immediately replace local currencies in those countries. During the transition period from 1999 to 2002, both traditional currencies and the euro will be used simultaneously for accounting purposes and non-cash transactions in each participating country. Euro cash will not be introduced until January 1, 2002, and by July 1, 2002 the traditional currencies will no longer be legal tender.

On the last day of 1998, the European Union set fixed exchange rates between the euro and the traditional national currency in each of the 11 adopting member nations. While the exchange rates within Euroland will remain fixed, exchange rates between the euro and non-euro countries will continue to vary freely and, in fact, several rates may be in use at one time (for example, actual and budgeted rates).

Reference

Currency Conversion Rules

The European Union has established the following rules for currency conversion:

- The exchange rate must be specified as a decimal value, r , with six significant digits (not six decimal places). For example, 123.456 has six significant digits but not six decimal places. This rate will establish the following relationship between the euro and the particular national currency:
 $1 \text{ euro} = r \text{ national units}$
- To convert from the euro to the national unit, multiply by r and round the result to two decimal places.
- To convert from the national currency to the euro, divide by r and round the result to two decimal places.
- To convert from one national currency to another, first convert from one national unit to the euro, rounding the result to at least three decimal places (your application rounds to exactly three decimal places). Then convert from the euro to the second national unit, rounding the result to two decimal places. This two-step conversion process is *triangulation*.

Example

Converting a Currency

The following example illustrates triangulation. In this case, 10 US dollars (USD) are converted to French francs (FRF). The exchange rate for USD to euros (EUR) is 1.17249. The exchange rate for FRF to euros is 6.55957.

- The 10 USD are converted to EUR by dividing the 10 USD by the EUR exchange rate of 1.17249:

$$\text{EUR} = 10 / 1.17249$$
 This results in 8.529 euros.
- The euros are converted to FRF by multiplying the above result by the exchange rate of FRF for euros (6.55957):

$$\text{FRF} = 8.529 * 6.55957$$
 The result is 55.96 FRF. This means 55.96 FRF are equivalent to 1 EUR.

Preparing for Currency Conversions

Although 11 or more currencies in the European Union will be converting to the euro, more than 100 currencies have a recognized status worldwide. In addition, you may need to define custom currencies for some applications.

You identify your currency codes and rates by creating a currency data source.

Creating the Currency Data Source

For each type of currency you need, you must supply the following values in your currency data source:

- A three-character code to identify the currency, such as USD for U.S. dollars or BEF for Belgian francs. (For a partial list of recognized currency codes, see Sample Currency Codes on page C-6.)
- One or more exchange rates for the currency.

There is no limit to the number of currencies you can add to your currency data source; the currencies you can define are not limited to official currencies and, therefore, the currency data source can be fully customized for your applications.

The currency data source can be any type of data source your application can access (for example, FOCUS, FIX, DB2, or VSAM). The currency Master File must have one field that identifies each currency ID you will use and one or more fields to specify the exchange rates.

We strongly recommend that you create a separate data source for the currency data rather than adding the currency fields to another data source. A separate currency data source enhances performance and minimizes resource utilization because FOCUS loads the currency data source into memory before you perform currency conversions.

Syntax**How to Create a Currency Data Source**

```

FIELD = CURRENCY_ID,,   FORMAT = A3,           ACTUAL = A3 , $
FIELD = rate_1,,       FORMAT = {D12.6|numeric_format1}, ACTUAL = A12,$
.
.
.
FIELD = rate_n,,       FORMAT = {D12.6|numeric_formatn}, ACTUAL = A12,$

```

where:

CURRENCY_ID

Is the required field name. The values stored in this field are the three-character codes that identify each currency, such as USD for U.S. dollars. Each currency ID can be a universally recognized code or a user-defined code.

Note: The code EUR is automatically recognized; you should *not* store this code in your currency data source. See Sample Currency Codes on page C-6 for a list of common currency codes.

rate_1...rate_n

Are types of rates (such as BUDGET, FASB, ACTUAL) to be used in currency conversions. Each rate is the number of national units that represent one euro.

numeric_format1...numeric_formatn

Are the display formats for the exchange rates. Each format must be numeric. The recommended format, D12.6, ensures that the rate is expressed with six significant digits as required by the European Union conversion rules. Do not use Integer format (I).

ACTUAL An

Is required only for non-FOCUS data sources.

Note: The maximum number of fields in the currency data source must not exceed 255 (that is, the CURRENCY_ID field plus 254 currency conversion fields).

Reference Sample Currency Codes

The following rates were in effect on December 31, 1998. Euroland countries as of that date are marked with an asterisk (*). Their rates are fixed and will not change; the rates for other countries can change over time:

Country	Currency Code	Rate
Austria*	ATS	13.7603
Belgium*	BEF	40.3399
Canada	CAD	1.7978
Denmark	DKK	7.46215
European Union	EUR	1
Finland*	FIM	5.94573
France*	FRF	6.55957
Germany*	DEM	1.95583
Greece	GRD	328.6
Ireland*	IEP	0.787564
Italy*	ITL	1936.27
Japan	JPY	133.149
Luxembourg*	LUF	40.3399
Netherlands*	NLG	2.20371
Norway	NOK	8.91039
Portugal*	PTE	200.482
Spain*	ESP	166.386
Sweden	SEK	9.52669
Switzerland	CHF	1.61093
UK	GBP	0.706739
USA	USD	1.17249

Example

Specifying Currency Codes and Rates in a Master File

The following Master File for a comma-delimited currency data source specifies two rates for each currency, ACTUAL and BUDGET:

```
FILE = CURRCODE, SUFFIX = COM,$
FIELD = CURRENCY_ID,, FORMAT = A3, ACTUAL = A3 , $
FIELD = ACTUAL, ALIAS = , FORMAT = D12.6, ACTUAL = A12 , $
FIELD = BUDGET, ALIAS = , FORMAT = D12.6, ACTUAL = A12 , $
```

The following is sample data for the currency data source defined by this Master File:

```
FRF, 6.55957, 6.50000,$
USD, 1.17249, 1.20000,$
BEF, 40.3399, 41.00000,$
```

Identifying Fields That Contain Currency Data

Once you have created your currency data source, you must identify the fields in your data sources that represent currency values. To designate a field as a currency-denominated value (a value that represents a number of units in a specific type of currency) add the CURRENCY attribute to one of the following:

- The FIELD specification in the Master File.
- The left side of a DEFINE or COMPUTE.

Syntax

How to Identify a Currency Value

Use the following syntax to identify a currency-denominated value

In a Master File

```
FIELD = currfield,, FORMAT = numeric_format, ..., CURR = {curr_id|codefield} , $
```

In a DEFINE in the Master File

```
DEFINE currfield/numeric_format CURR curr_id = expression ; $
```

In a DEFINE FILE command

```
DEFINE FILE filename  
currfield/numeric_format CURR curr_id = expression ;  
END
```

In a COMPUTE command

```
COMPUTE currfield/numeric_format CURR curr_id = expression ;
```

where:

currfield

Is the name of the currency-denominated field.

numeric_format

Is a numeric format. Depending on the currency denomination involved, the recommended number of decimal places is either two or zero. Do not use I or F format.

CURR

Indicates that the field value represents a currency-denominated value. CURR is an abbreviation of CURRENCY, which is the full attribute name.

curr_id

Is the three-character currency ID associated with the field. In order to perform currency conversions, this ID must either be the value EUR or match a CURRENCY_ID value in your currency data source.

codefield

Is the name of a field, qualified if necessary, that contains the currency ID associated with *currfield*. The code field should have format A3 or longer and is interpreted as containing the currency ID value in its first three bytes. For example:

```
FIELD = PRICE,, FORMAT = P12.2C, ..., CURR = TABLE.FLD1,$
.
.
.
FIELD = FLD1,, FORMAT = A3, ..., $
```

The field named FLD1 contains the currency ID for the field named PRICE.

filename

Is the name of the file for which this field is defined.

expression

Is a valid expression.

Example

Identifying a Currency-Denominated Field

Assume that the currency data source contains the currency ID value BEF (Belgian francs).

If the FINANCE data source contains a field named PRICE that is denominated in Belgian francs, the description of PRICE in the FINANCE Master File could be:

```
FIELD = PRICE, ALIAS=, FORMAT = P17.2, CURR=BEF,$
```

Activating the Currency Data Source

Before you can perform currency conversions, you must bring the relevant currency data source into memory by issuing the SET EUROFILE command.

Issue the EUROFILE command in a FOCEXEC or any supported profile.

Syntax

How to Activate Your Currency Data Source

```
SET EUROFILE = {ddname|OFF}
```

where:

ddname

Is the name of the Master File for the currency data source. There is no default value for EUROFILE. The ddname must refer to a data source known to and accessible by your application in read-only mode.

OFF

Deactivates the currency data source and removes it from memory.

If you want to access a different currency data source, you can re-issue the SET EUROFILE command.

Reference

Restrictions for Issuing the SET EUROFILE Command

The following restrictions apply to the issuing of the SET EUROFILE command:

- You cannot append any additional SET parameters to the SET EUROFILE command line. For example, the PAUSE setting would be lost if you issued the following command:

```
SET EUROFILE=filename , PAUSE=OFF
```

- You cannot issue the SET EUROFILE command within a TABLE request.

Querying the Currency Data Source in Effect

You can issue a query to determine what currency data source is in effect. To do this, issue the ? SET ALL query command or the ? EUROFILE query command.

Syntax **How to Determine the Currency Data Source in Effect**

? SET EUROFILE

Example **Determining the Currency Data Source in Effect**

Assume the currency data source is named CURRCODE. Issuing the command

? EUROFILE

produces the following information:

EUROFILE	CURRCODE
-----------------	-----------------

Reference **SET EUROFILE Error Messages and Notes**

- Issuing the SET EUROFILE command when the currency data source Master File does not exist generates the following error message:
(FOC205) THE DESCRIPTION CANNOT BE FOUND FOR FILE NAMED: *ddname*
- Issuing the SET EUROFILE command when the currency Master File specifies a FOCUS data source and the associated FOCUS data source does not exist generates the following error message:
(FOC036) NO DATA FOUND FOR THE FOCUS FILE NAMED: *name*

Processing Currency Data

After you have created your currency data source, identified the currency-denominated fields in your data sources, and activated your currency data source, you can perform currency conversions.

Each currency ID in your currency data source generates a virtual conversion function whose name is the same as its currency ID. For example, if you added BEF to your currency data source, a virtual BEF currency conversion function will be generated.

The euro function, EUR, is supplied automatically with your application. You do not need to add the EUR currency ID to your currency data source.

Syntax

How to Convert Currency Data

Use the following syntax for calling a currency conversion function

- In a TABLE, GRAPH, or MODIFY (WebFOCUS Windows version only) procedure:

```
DEFINE FILE filename
  result/format [CURR curr_id] = curr_id(infield, rate1 [,rate2]);
END
```

or

```
COMPUTE result/format [CURR curr_id] = curr_id(infield, rate1
  [,rate2]);
```

- In a Master File:

```
DEFINE result/format [CURR curr_id] = curr_id(infield, rate1
  [,rate2]);$
```

where:

filename

Is the name of the file for which this field is defined.

result

Is the converted currency value.

format

Must be a numeric format. Depending on the currency denomination involved, the recommended number of decimal places is either two or zero. Do not use I or F format. The result will always be rounded to two decimal places, which will display if the format allows at least two decimal places.

curr_id

Is the currency ID of the result field. This ID must be the value EUR or match a currency ID in your currency data source; any other value generates the following message:

```
(FOC263) EXTERNAL FUNCTION OR LOAD MODULE NOT FOUND: curr_id
```

Note: The CURR attribute on the left side of the DEFINE or COMPUTE identifies the result field as a currency-denominated value which can be passed as an argument to a currency function in subsequent currency calculations. Adding this attribute to the left side of the DEFINE or COMPUTE does not invoke any format or value conversion on the calculated result.

infield

Is a currency-denominated value. This input value will be converted from its original currency to the *curr_id* denomination. If the *infield* and *result* currencies are the same, no calculation is performed and the *result* value is the same as the *infield* value.

rate1

Is the name of a rate field from the currency data source. The *infield* value is divided by its currency's *rate1* value to produce the equivalent number of euros.

If *rate2* is not specified in the currency calculation and triangulation is required, this intermediate result is then multiplied by the *result* currency's *rate1* value to complete the conversion.

In certain cases, you may need to provide different rates for special purposes. In these situations you can specify any field or numeric constant for *rate1* as long as it indicates the number of units of the *infield* currency denomination that equals one euro.

rate2

Is the name of a rate field from the currency data source. This argument is only used for those cases of triangulation in which you need to specify different rate fields for the *infield* and *result* currencies. It is ignored if the euro is one of the currencies involved in the calculation.

The number of euros that was derived using *rate1* is multiplied by the *result* currency's *rate2* value to complete the conversion.

In certain cases, you may need to provide different rates for special purposes. In these situations you can specify any field or numeric constant for *rate2* as long as it indicates the number of units of the *result* currency denomination that equals one euro.

Example

Converting Currencies

Assume that the currency data source contains the currency IDs USD and BEF, and that PRICE is denominated in Belgian francs as follows:

```
FIELD = PRICE, ALIAS=, FORMAT = P17.2, CURR=BEF,$
```

- The following example converts PRICE to euros and stores the result in PRICE2 using the BUDGET conversion rate for the BEF currency ID:

```
COMPUTE PRICE2/P17.2 CURR EUR = EUR(PRICE, BUDGET);
```

- This example converts PRICE from Belgian francs to US dollars using the triangulation rule:

```
DEFINE PRICE3/P17.2 CURR USD = USD(PRICE, ACTUAL);$
```

First PRICE is divided by the ACTUAL rate for Belgian francs to derive the number of euros rounded to three decimal places. Then this intermediate value is multiplied by the ACTUAL rate for US dollars and rounded to two decimal places.

- The following example uses a numeric constant for the conversion rate:

```
DEFINE PRICE4/P17.2 CURR EUR = EUR(PRICE,5);$
```

- The next example uses the ACTUAL rate for Belgian francs in the division and the BUDGET rate for US dollars in the multiplication:

```
DEFINE PRICE5/P17.2 CURR USD = USD(PRICE, ACTUAL, BUDGET);$
```

Reference

Currency Calculation Processing and Messages

The result is always calculated with very high precision, 31 to 36 significant digits, depending on platform. The precision of the final result is always rounded to two decimal places. In order to display the result to the proper precision, its format must allow at least two decimal places.

Issuing a TABLE request against a Master File that specifies a currency code not listed in the active currency data source generates the following message:

```
(FOC1911) CURRENCY IN FILE DESCRIPTION NOT FOUND IN DATA
```

A syntax error or undefined field name in a currency conversion expression generates the following message:

```
(FOC1912) ERROR IN PARSING CURRENCY STATEMENT
```

Example**Converting U.S. Dollars to Euros, French Francs, and Belgian Francs**

Assume PRICE is denominated in U.S. dollars and ACTUAL is the name of a rate in the currency data source. Using the currency conversion rates from *Sample Currency Codes* on page 11-6, the following FOCEXEC converts PRICE to euros, French francs, and Belgian francs:

```

-* CURRCODE IS THE CURRENCY DATABASE
-* CURRDATA IS THE DATA SOURCE WITH CURRENCY-DENOMINATED FIELDS
-* THE FOLLOWING ALLOCATIONS ARE FOR RUNNING UNDER MVS
-* DYNAM ALLOC FILE CURRCODE DA USER1.FOCEXEC.DATA(CURRCODE) SHR REU
-* DYNAM ALLOC FILE CURRDATA DA USER1.FOCEXEC.DATA(CURRDATA) SHR REU

-* THE FOLLOWING ALLOCATIONS ARE FOR RUNNING UNDER WINDOWS
-* FILEDEF CURRCODE DISC C:\IBI\APPS\GGDEMO\CURRCODE.COM
-* FILEDEF CURRDATA DISC C:\IBI\APPS\GGDEMO\CURRDATA.COM

```

```
SET EUROFILE = CURRCODE
```

```

DEFINE FILE CURRDATA
PRICEEUR/P17.2 CURR EUR = EUR(PRICE, ACTUAL);
END

```

```

TABLE FILE CURRDATA
PRINT PRICE PRICEEUR AND COMPUTE
PRICEFRF/P17.2 CURR FRF = FRF(PRICE, ACTUAL);
PRICEBEF/P17.2 CURR BEF = BEF(PRICE, ACTUAL);
END

```

This request generates the following report:

<u>PRICE</u>	<u>PRICEEUR</u>	<u>PRICEFRF</u>	<u>PRICEBEF</u>
5.00	4.26	27.97	172.01
6.00	5.12	33.57	206.42
40.00	34.12	223.78	1376.20
10.00	8.53	55.95	344.06

Note: You cannot use the derived euro value PRICEEUR in a conversion from USD to BEF. PRICEEUR has two decimal places (P17.2), not three, as the triangulation rules require.

Punctuating Large Numbers

Countries differ in how they punctuate numbers. You can reflect these differences in your reports using Continental Decimal Notation (CDN). CDN specifies the punctuation used in numerical notation. CDN is supported for output in report requests but is not supported in DEFINE or COMPUTE commands. The CDN parameter also determines the punctuation of numbers affected by the CENT-ZERO parameter. The option chosen for CDN determines the character used to separate decimal places affected by the CENT-ZERO parameter.

Syntax

How to Determine the Punctuation of Large Numbers

```
SET CDN = option
```

where:

option is one of the following:

OFF

Separates thousands with a comma, and fractions with a period. For example, the number 3,045,000.76 is represented as 3,045,000.76. This format is preferred in the United States, Canada, Mexico, and the United Kingdom. This value is the default.

ON

Separates thousands with a period, and fractions with a comma. For example, the number 3,045,000.76 is represented as 3.045.000,76. This format is preferred in Germany, Denmark, Italy, Spain, and Brazil

Example

Determining the Punctuation of Large Numbers

The following request punctuates numbers using a period to separate thousands, and a comma to separate fractions.

```
SET CDN = ON
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME SALARY
END
```

The output is:

<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>SALARY</u>
STEVENS	ALFRED	\$11.000,00
SMITH	MARY	\$13.200,00
JONES	DIANE	\$18.480,00
JONES	DIANE	\$17.750,00
BANNING	JOHN	\$29.700,00
IRVING	JOAN	\$26.862,00
IRVING	JOAN	\$24.420,00
ROMANS	ANTHONY	\$21.120,00
MCCOY	JOHN	\$18.480,00
BLACKWOOD	ROSEMARIE	\$21.780,00
MCKNIGHT	ROGER	\$16.100,00
MCKNIGHT	ROGER	\$15.000,00
CROSS	BARBARA	\$27.062,00
CROSS	BARBARA	\$25.775,00

Accessing Local Language Error Messages

WebFOCUS contains a set of error messages for six different languages. To specify the set of error messages, use the LANGUAGE parameter. If you use the automated WebFOCUS installation for Windows NT or UNIX, the local language error messages are automatically accessed.

Syntax

How to Set the Local Language

```
SET LANG[UAGE] = {value|ENGLISH}
```

where:

value

Determines the set of local language error messages to be accessed. You can use the language name or abbreviation to specify the local language. Possible values are:

Name	Abbreviation	Description
<u>ENGLISH</u>	AME	Uses the English set of error messages.
<u>DUTCH</u>	DUT	Uses the Dutch set of error messages.
<u>FRENCH</u>	FRE	Uses the French set of error messages.
<u>GERMAN</u>	GER	Uses the German set of error messages.
<u>JAPANESE</u>	JPN	Uses the Japanese set of error messages.
<u>SPANISH</u>	SPA	Uses the Spanish set of error messages.

Note:

- On EBCIDC based platforms, the LANG and CODE_PAGE settings must be compatible. For example, if LANG=FRE, then CODE_PAGE must be set to the corresponding code page for displaying French characters.
- Certain error message text will always appear in English.

APPENDIX D

Bibliography

NLS Concepts and Principles of International Computing

O'Donnell, Sandra Martin, *Programming for the World: A Guide to Internationalization* (Englewood Cliffs: PTR Prentice Hall, 1994). ISBN: 0-13-722190-8

Lunde, Ken, *CJKV Information Processing* (Sebastopol: O'Reilly, 1999). ISBN: 1-56592-224-7

History of Code Pages

Searle, Steven J., "A Brief History of Character Codes in North America, Europe, and East Asia." <http://tronweb.super-nova.co.jp/characcodehist.html>

Windows and DOS Code Pages

Kano, Nadine, *Developing International Software For Windows 95 and Windows NT* (Redmond: Microsoft Press, 1995). ISBN: 1-55615-840-8

Note: This book is currently out of print but is available in the Microsoft Developers Network (MSDN) documentation.

Microsoft Corporation, *User's Guide: Microsoft MS-DOS 6*, Chapter 9, "Customizing for International Use."

UNIX Code Pages

Uren, Emmanuel, Robert Howard, and Tiziana Perinotti, *Software Internationalization and Localization: An Introduction* (New York: Van Nostrand Reinhold, 1993), Appendix J. ISBN: 0-442-01498-8

EBCDIC Code Pages

IBM National Language Technical Center, *National Language Support Reference Manual*, Volume 2 (1992).

Unicode

Graham, Tony, *Unicode: A Primer* (Foster City: M&T Books, 2000). ISBN: 0-7645-4625-2

Unicode Consortium, *The Unicode Standard: Version 3.0* (Reading, Mass.: Addison-Wesley, 2000). ISBN: 0-201-61633-5

Periodicals and Web Sites

MultiLingual Computing & Technology [bimonthly magazine] Publisher: MultiLingual Computing, Inc., 319 North First Avenue, Sandpoint, Idaho 83864, USA. TEL: (208) 263-8178

MultiLingual Computing Web site: <http://www.multilingual.com>

Unicode Consortium Web site: <http://www.unicode.org>

Glossary

This glossary includes definitions and helpful information about terms used in this documentation. It may include terms pertaining to National Language Support (NLS) that are not used in this documentation.

alphanumeric	A computer data type for handling text.
ANSI	Acronym for American National Standards Institute. It is used to refer to the Microsoft Windows 1252 Windows Latin-1 code page for IBM compatible personal computers.
ASCII	Acronym for American Standard Code for Information Interchange. It is a 7-bit code set that is made up of 96 upper and lowercase English letters, numerals 0-9, punctuation marks, and mathematical symbols. It does not include the additional national characters required by other European languages.
ASCII-transparent	A code set or code page where values in the range 0x20-0x7F represent the 96 7-bit ASCII characters. Most code pages are ASCII-transparent with the exception of the IBM EBCDIC set.
BIDI	Abbreviation (pronounced “buy-die”) of bidirectional, in reference to mixed left-to-right and right-to-left text. Hebrew and Arabic are known as the BIDI languages because text normally reads from right-to-left, except for embedded numerals, and words and abbreviations in Latin script, which are read from left-to-right.
Big Five	A traditional Chinese character set and encoding method used in Taiwan. The encoding method is modeled on, and is very similar to, Japanese Shift-JIS. It supports the larger number of ideographs used in written Chinese, as opposed to the smaller set of kanji ideographs used in written Japanese.

binary	<p>A system of numerical notation to the base two. (The decimal system uses base ten.)</p> <p>The characters 0 and 1 represent binary digits, and computer systems store and process data, including text data, in binary form. However, the code point values of specific text characters are usually expressed in hexadecimal notation, not binary.</p>
browser	<p>A software utility for navigating and accessing information on the World Wide Web (Internet) or an intranet. A browser interprets hypertext markup language (HTML) and displays it as graphic information. Familiar examples are Netscape Navigator and Microsoft Internet Explorer.</p>
case	<p>The capitalized (uppercase) or uncapitalized (lowercase) form of alphabetic characters, usually in Latin script. Some languages do not have case such as Chinese, Japanese, Korean, and the Indic languages used on the Indian subcontinent.</p>
case conversion	<p>Converting a letter from its lowercase to uppercase form (or vice versa). For computer systems, this is sometimes called <i>monocasing</i>. The NLS API has a facility that handles case conversion.</p>
character	<p>A written symbol that can be combined with other symbols to represent text. Examples include a letter, a letter with a diacritic, an ideograph, a numeral, or a punctuation mark. Also used to refer to a Chinese ideograph (Chinese character).</p>
character set	<p>A set of characters that may consist of letters, ideographs, numerals, symbols, and punctuation marks. A character set typically includes all the written symbols required by one or more related languages.</p>
Chinese characters	<p>The ideographs, or <i>hanzi</i>, used to write the Chinese language. There are two forms of Chinese characters: traditional characters (more complex), which are used in Taiwan, Hong Kong, and by overseas Chinese communities, and simplified characters, which are used in the People's Republic of China.</p>

CJK	Acronym for Chinese/Japanese/Korean. Computer systems use the same strategy for text processing with these three languages. Double-byte DBCS codes are used to create code sets that include the thousands of ideographs and written symbols used in each language.
client	<p>In the client/server computing architecture, client applications typically host a user tool or other program for data manipulation and access information on remote servers or host computers. Often the client application runs on a desktop or personal computer.</p> <p>For Information Builders' products, client can refer to any of the following components connecting to an EDA Server:</p> <ul style="list-style-type: none">• The WebFOCUS CGI or ISAPI client.• WebFOCUS (Windows version).• A third party product.
client/server computing	<p>A computing architecture in which client applications, often running on desktop or personal computers, access information on remote servers or host computers. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized, multi-user functionality.</p> <p>Client/server computing products, such as WebFOCUS and EDA, generally have the client and server software installed on different computers. These computers can have different operating system platforms. Since code pages are platform specific, the client and server components will use different code pages, even when processing the same language.</p> <p>If client and server are using different code pages, a transcoding service must convert the numeric values for specific letters from one code page to the other.</p>
code page	A code page is the computer representation of a character set. Each written symbol in a language or script is assigned a unique number, usually expressed in hexadecimal notation. Code pages usually have a unique identifying number and are language and platform specific.

code page 137	Information Builders' proprietary code page, which handles all major North American, South American, and Western European languages (except Greek) for Windows NT and UNIX. Code page 137 is functionally equivalent to Microsoft Windows 1252 and UNIX ISO 8859-1 code pages.
code point	The numeric value assigned to a specific value in a code set. For example, the letter 'A' has a code value of 0x41 (hexadecimal notation) in 7-bit ASCII. Also called <i>code value</i> or <i>code position</i> .
code set	An unambiguous mapping of the members of a character set to specific numeric values.
CDRA	Acronym for IBM's Character Data Representation Architecture. The graphic character representation standards (code page numbers, layouts, and the binary values for specific written symbols) used by Information Builders are in accordance with CDRA.
DBCS	Acronym for Double-Byte Character Set. It refers to the double-byte (16-bit) character encodings used to represent East Asian writing systems (Chinese, Japanese, and Korean). The NLS API can be configured to handle these scripts.
decimal	A system of numerical notation to the base ten. In the familiar decimal notation, the Arabic numerals 0-9 represent the digits.
diacritic	A mark added to a letter that usually provides information about pronunciation or the stress that should be given to a syllable. Examples include acute (´) and grave (`) accents, and diaereses (¨). In some languages, certain letters with diacritics are considered separate letters of the alphabet such as the Swedish national characters Å/å, Ä/ä, Ö/ö.
EBCDIC	Acronym for Extended Binary Coded Decimal Interchange Code. A family of code pages used on IBM mainframes and other IBM operating systems such as OS/400 and MVS/OE.

EBCDIK	Acronym for Extended Binary Coded Decimal Interchange Kana Code. A Japanese version of IBM's 8-bit SBCS EBCDIC code page family that includes uppercase Roman letters, numerals, symbols, half-width katakana, and control characters. Note that EBCDIK has neither lowercase Roman letters, nor hiragana. The half-width katakana characters occupy the code points for lowercase letters in the other EBCDIC code pages.
EDA	Acronym for Enterprise Data Access. A family of client/server products developed by Information Builders. EDA enables different products to work together to provide access to relational and non-relational data on interconnected multi-vendor systems.
EDA Client	The middleware that manages the data flow between the client application and the server on behalf of the client.
EDA Hub Server	A special EDA Server that can have one or more EDA Servers (or EDA Subservers) as its data sources. For NLS, EDA Subservers with data sources using different code pages can be joined with EDA Hub Servers.
EDA Server	Server software that manages the reading and writing of data from multiple sources in response to requests from multiple client applications. The NLS API is a core component of the EDA Server and supports all aspects of international computing such as transcoding, case conversion, and data formatting.
EDA Subserver	A special EDA Server whose client application is an EDA Hub Server. (The EDA Subserver is the EDA Hub Server's data source.) For NLS, the Subserver and Hub Server may (by design) be using different code pages.

encoding	A scheme or strategy for assigning specific numeric values to represent particular written symbols. See also <i>code set</i> .
EUC	Extended UNIX Code. Japanese encoding method in two forms: <ul style="list-style-type: none">• EUC complete two-byte format (fixed-width 16-bit encoding), the most common Japanese implementation for UNIX.• EUC packed format (variable-width encoding, that is, uses modal or stateful encoding).
extended characters	<p>The single-byte SBCS non-English European letters found in the range 0x80-0xFF on DOS, Windows, UNIX, and other ASCII-transparent code pages. Examples are German umlauts (Ä/ä, Ö/ö, Ü/ü) and Eszett (ß), French accented characters (á, à, â), and special Swedish letters (Å/å, Ä/ä, Ö/ö).</p> <p>These characters are called extended characters because in the DOS code page architecture for the original IBM PC (1981), these were in the extended space 0x80-0xFF above the base area 0x20-0x7E reserved for American English letters, numerals, and punctuation.</p> <p>For the DOS and ANSI (Windows and UNIX) code page architecture, all national characters are extended characters.</p>
filter	A computer mechanism, such as Information Builders' NLS API, that handles transcoding of text data between code pages. The NLS API changes the numeric value (in hexadecimal notation) that represents each written character in a data stream so that the text can be displayed properly on a linked computer using a different operating system code page for the same language or script.

fixed-width encoding	One of the three character encoding strategies for the CJK (Chinese, Japanese, and Korean) DBCS languages. Fixed-width encoding methods use the same number of bytes to represent all the characters available in a character set. There is no switching between one and two-byte-per-character modes. This encoding method simplifies searching, indexing, and sorting of text, but can use up a lot of space. Examples of fixed-width encoding methods are ASCII (7 or 8-bit forms), EUC complete two-byte format, and Unicode UTF-16.
font	A collection of glyphs (images) used for the visual depiction of character data. Computer systems typically have mechanisms for displaying fonts on a screen and printing them.
GB	A simplified Chinese character set and encoding used in the People's Republic of China. GB is the abbreviation for the Chinese words "National Standard."
GCGID	Acronym for Graphic Character Global Identifier. Refer to IBM's CDRA Library documentation set (SC09-9391-00 Registry) for more information on GCGIDs.
graphic character	<p>Another term for a character used in IBM and other documentation.</p> <p>A written symbol that can be combined with other symbols to represent text. Examples include a letter, a letter with diacritic, an ideograph, a digit, or a punctuation symbol. Also used to refer to a Chinese ideograph (Chinese character).</p>
hangul	The native Korean phonetic script. Consonants and vowels (jamo) are grouped into square-like symbols, each of which represents a single syllable. These symbols are not ideographs (Korean versions of the Chinese ideographs are called hanja). They are encoded with the same DBCS techniques as Japanese kanji and Chinese characters.

hanja	<p>Korean ideographs. The Korean versions of the ideographs originally borrowed from the Chinese writing system, and used to write the completely unrelated Korean spoken language.</p>
hexadecimal	<p>A system of numerical notation to the base sixteen. (The decimal system uses base ten.)</p> <p>In standard hexadecimal notation, the characters 0-9 and A-F represent the numeric values 0 to 15(F). Specific hex numbers have the form 0x4F, where 0x is a prefix indicating a hex number, and 4F are hex digits. Code point values are usually expressed as hex numbers.</p>
hiragana	<p>A cursive (rounded) native Japanese phonetic script usually used for writing words of native Japanese origin, or for grammatical particles. Together with katakana is collectively called <i>kana</i>.</p>
HTML	<p>Hypertext Markup Language. A simple markup language used to create hypertext documents that are portable from one platform to another. HTML files are plain text files with codes embedded (indicated by markup tags) to indicate formatting and hypertext links. This is the formatting language used for documents on the World Wide Web.</p>
ideograph	<p>A character representing an idea (or meaning) in contrast to a sound (or pronunciation). Some writing systems, including Chinese and Japanese, use thousands of ideographs.</p>
IME	<p>Acronym for Input Method Editor. Software for facilitating the input of the Far Eastern CJK (Chinese, Japanese, and Korean), and other languages.</p> <p>There are several standard input methods for each of the CJK languages. An IME utility is typically included with the computer operating system, and operating systems often support several of the standard input methods. Other languages not currently supported by Information Builders, such as the Indic languages of the Indian Subcontinent, also use IMEs for input.</p>

Internet	Abbreviation for internetwork. A set of dissimilar computer networks joined together by gateways that handle data transfer and the conversion of messages from the sending network to the protocols used by the receiving networks. These networks and gateways use the TCP/IP suite of protocols. The World Wide Web (or Web) is an important facility that runs on the Internet.
ISO	Acronym for the French name of the International Organization for Standardization. It is sometimes used to refer to the ISO-8859-x family of UNIX code pages, and specifically the ISO-8859-1 Latin-1 code page, which has the same layout as the Windows 1252 code page and Information Builders' proprietary code page 137.
JIS	Acronym for Japanese Industrial Standard. A computer or other industry standard established by JISC, the Japanese Industrial Standards Committee.
join	An SQL or database management facility that allows separate physical data sources to be combined as a single (larger) logical database using key values in common fields.
kana	Collective name for the two native Japanese phonetic scripts: <i>hiragana</i> and <i>katakana</i> .
kanji	Japanese ideographs. The Japanese versions of the ideographs originally borrowed from the Chinese writing system, and used to write the completely unrelated Japanese spoken language.
katakana	The angular native Japanese phonetic script, most often used for writing words of foreign origin or interjections. Together with hiragana is collectively called <i>kana</i> .
KS X 1001:1992	A Korean character set encoding method, which supports the Korean hangul phonetic system, along with the less often used hanja ideographs of Chinese origin. Before 1997, KS X 1001:1992 was known as KS C 5601:1992, although the encoding method remains the same.

language	Spoken language is the words, their pronunciation, and the methods of combining them used and understood by a community. Written language uses symbols from a script to transcribe the spoken language. Examples of spoken and written languages are English, German, French, Swedish, and Japanese.
locale features	<p>Computer hardware or software features that are dependent and need to be changed based on language, country, and cultural convention. Examples are proper sort order, computer keyboard layout, expected fonts, and date, time, number, and currency formats.</p> <p>For example, the number 1,234,567.89 in the U.S. should be expressed as 1.234.567,89 in Germany, and as 1 234 567,89 in France (note that spaces separate thousands).</p> <p>Without locally expected locale features, a software product, particularly a consumer product, is badly flawed.</p>
modal encoding	One of the three character encoding strategies for the CJK DBCS languages (Chinese, Japanese, and Korean). Modal encoding methods use a two-stage encoding process. The first stage is mode switching initiated by the escape sequence. The second stage is the handling of the actual bytes that represent the characters. Modal encoding methods typically use seven-bit codes. An example is Japanese JIS encoding.
monocasing	A computer term for <i>Case Conversion</i> . Converting a letter from its lowercase to uppercase form (or vice versa). The NLS API has a facility that handles monocasing.
national characters	Non-English written symbols such as European letters or extended characters. Examples are German umlauts (Ä/ä, Ö/ö, Ü/ü) and Eszett (ß), French accented characters (á, à, â), and special Swedish letters (Å/å, Ä/ä, Ö/ö). Other examples include Asian language characters such as Japanese kanji and kana, and traditional and simplified Chinese characters, or Korean hangul.

NLS	<p>Acronym for National Language Support. Providing customization of hardware and software to enhance usability in that country or language.</p> <p>For Information Builders' products, the most important part of NLS is seamlessly interpreting all national characters embedded in any data source. NLS also includes correct sorting, upper/lower case conversion, and locally expected data formats (time, date, currency, and numbers).</p>
NLS API	<p>Information Builders' proprietary facility for supporting international computing. Its primary function is the transcoding of data on a character-by-character basis, between the code page of the server component and the code page of the client component. It is a core component of Information Builders' EDA Server that is built into WebFOCUS and most other Information Builders' products.</p>
NLV	<p>Acronym for National Language Version. NLV is a version of a software product where the entire user interface (menus, dialog boxes, screen text, online help systems error messages, and utility screens) is in a particular language.</p> <p>Examples of NLV products at Information Builders are the French, German, Japanese, and Spanish editions of WebFOCUS. Information Builders' NLV products include both the translated and localized user interface, as well as handling of local language data.</p>
non-modal encoding	<p>One of the three character encoding strategies for the CJK DBCS languages (Chinese, Japanese, and Korean). Non-modal encoding methods make use of the numeric value of a text stream's bytes for deciding when to switch between one and two-byte-per character modes. These encoding methods typically use eight-bit codes. Examples are Japanese Shift-JIS for IBM compatible PC's and EUC packed format for UNIX.</p>
OEM	<p>Acronym for Original Equipment Manufacturer. For NLS, OEM refers to the MS/PC DOS family of code pages for IBM-compatible personal computers.</p>

radix character	The symbol that separates whole and fractional quantities in a floating point number. For example, the number 1,234,567.89 in the U.S. (dot is the radix character) is expressed as 1.234.567,89 in Germany (comma is the radix character).
romaji	Japanese name for Latin script, which, along with kanji, hiragana, and katakana, is an integral part of the Japanese written language. It is used for numerical tables, abbreviations (for example WTO and UNESCO), and transliteration and transcription. Standard Japanese encodings, like Shift-JIS for IBM compatible personal computers, and EUC for UNIX, support the 7-bit ASCII (English) characters, but do not include the extra national characters required to write most other European languages.
SBCS	Acronym for Single-Byte Character Set. It refers to the single-byte (8-bit) encodings used to represent alphabets such as Eastern and Western European languages, Greek, Cyrillic (Russian), Arabic, and Hebrew.
script	<p>A script is a collection of symbols used to represent textual information in a writing system. These symbols might be letters of the alphabet, the numerals 0-9, punctuation marks, and mathematical symbols.</p> <p>Scripts are the world's major writing systems, which include Latin script (Roman letters), Greek script, Cyrillic script (Russian), and Japanese script.</p>
server	<p>In the client/server computing architecture, server applications typically run on centralized computers and provide information in response to remote requests from client applications or other computers.</p> <p>When referring to the Information Builders family of products, server refers specifically to an EDA Server.</p>
Shift-JIS	The Japanese encoding method used for IBM compatible personal computers. A non-modal encoding that was given this name because the code point ranges for the first identifying byte for double-byte DBCS kanji "shift around" the ranges for single-byte SBCS ASCII/JIS Roman and half-width katakana. Also called MS Kanji.

Simplified Chinese	The streamlined form of the written Chinese characters developed and used in the People's Republic of China. The traditional forms of many characters were simplified, so they could be written with fewer strokes (lines). This was intended to make it easier to learn to read and write Chinese. Many of the simplified forms are actually standard cursive versions of specific characters used for centuries to write rapidly as in notes or letters. Many other characters are exactly the same in the traditional and simplified sets.
stateful encoding	<p>A character encoding method that contains sequences embedded in the data that describe how to interpret the data. For example, special escape sequences (sets of specific binary codes beginning with ESC) designate which code set the data following it belongs to.</p> <p>While stateful encoding was developed to allow multiple scripts in a datastream (such as Greek and English), it is primarily used for the CJK languages like Japanese, which mix DBCS kanji (ideographs of Chinese origin) with three other SBCS scripts (Roman letters, hiragana, and katakana). Stateful encoding is essentially the same as modal encoding.</p>
thousands separator	The symbol used to separate groups of thousands in a written decimal number. For example, the number 1,234,567.89 in the U.S. (comma is the thousands separator) is expressed as 1.234.567,89 in Germany (dot is the thousands separator), and as 1 234 567,89 in France (space is the thousands separator).
Traditional Chinese	The original, more complex written forms of Chinese characters used in Taiwan, Hong Kong, and overseas Chinese communities. Simplified Chinese characters are used in the People's Republic of China.
transcoding	Conversion of character data between different character sets. A transcoding mechanism, or filter, like Information Builders' NLS API, can change the numeric value (in hexadecimal notation) that represents each written character so that it can be displayed properly on a linked computer using a different operating system code page for the same language or script.

UCS	Acronym for Universal Character Set. The term for Unicode used in the ISO/IEC 10646 standard that defines the encoding system.
Unicode	An international standard for character encoding that includes every written symbol in all the world's living languages. The two common Unicode implementations for computer systems are UTF-16, which is a fixed-width, two-byte encoding, and UTF-8, which is a variable length encoding, where each written symbol is represented by a one-to-four byte code.
UTF	Acronym for Unicode (or UCS) Transformation Format. UTF-x is one of the standard ways of encoding Unicode values. UTF-8 and UTF-16 are the most important, but there are additional formats for big-endian, little-endian, 7-bit e-mail systems, and other special purposes.
UTF-8	<p>Acronym for Unicode (or UCS) Transformation Format, 8-bit encoding form. A variable length standard for encoding Unicode values, where each written symbol is represented by a one-to-four byte code.</p> <p>In general, ASCII/ANSI Latin letters are one byte, symbols from other alphabets, like Greek, are two bytes, CJK East Asian characters are three bytes, and rarely used symbols (called surrogates) are four bytes. These codes are based on an algorithm, which makes it easy for applications to parse UTF-8 text, even though individual symbols can be one-to-four bytes in length.</p> <p>UTF-8 is widely implemented in browser, database, and other current IT implementations. Since UTF-8 is ASCII/ANSI transparent (Latin letters are one byte), it has performance and other technical advantages over UTF-16.</p>
UTF-16	Acronym for Unicode (or UCS) Transformation Format, 16-bit encoding form. A fixed-width standard for encoding Unicode values, where each written symbol is represented by a two-byte code. In this encoding, the encoded value (in hex) is the same as the Unicode code point value.

ZAP

A utility or mechanism that is used to make minor technical changes in program code.

For NLS, it is also a command option for the Transcoding Services Generation Utility (TSGU) to change transcoding values for specific code points in transcoding tables.

Index

Symbols

? EUROFILE command, C-11

? LANG command, 6-17

A

alphanumerics, 7-5, G-1

ANSI (American National Standards Institute) code page, B-2, G-1

AnsiOem setting, 6-5

ASCII code set, B-2, G-1

ASCII-transparent code set, 6-2, G-1

B

BIDI (bidirectional text), G-1

Big Five Chinese character set, G-1

binary numerical notation, G-2

browsers, G-2

C

case conversion, 4-12, G-2

cases, G-2

CASETBL file, 5-12

CDN (Continental Decimal Notation), C-16
punctuating large numbers, C-16

CDRA (Character Data Representation), G-4

Character Data Representation (CDRA), G-4

character sets, 1-3, G-2

 DBCS, 1-3

 SBCS, 1-3

characters, G-2

Chinese characters, G-2

Chinese/Japanese/Korean (CJK) scripts, B-3, G-3

CJK (Chinese/Japanese/Korean) scripts, B-3, G-3

client and server code pages, 1-7
 transcoding, 1-7

client and server configuration, 6-13
 accessing ANSI-based data, 6-14
 accessing OEM (DOS)-based data, 6-15
 default settings, 6-14

client applications, G-3

Client NLS, 3-9
 configuring, 3-9
 customizing services, 4-17

code page 137, G-4

code page definition files, 4-14, 4-16, 5-14, 5-16

code page environments, 7-5

code page generation list files, 4-4, 4-5, 4-8, 5-5
 adding alternate code pages, 5-5

code page parameters, 6-16
 Master Files, 6-16

code page settings, 6-8
 DOS, 6-8
 Windows, 6-8

code pages, 1-3, 1-4, 3-2, 4-8, B-1, G-3
 adding alternates, 4-8
 changing settings, 4-9
 combining data, 1-14
 combining scripts with Unicode, 1-16, 7-5
 definition files, 4-14, 4-16
 history, B-1, B-2
 identifying alternates, 4-5
 international languages and, 6-16
 key code pages, 1-5
 reading data in different languages, 1-12
 selecting, B-4
 selecting languages, 3-2, 3-7, 3-9, 3-11
 transcoding tables, 4-8

code pages (*continued*)
 transcoding with multiple client code pages, 1-10

code points, 1-6, G-4

code sets, G-4

communications configuration files, 4-17, 4-18, 5-17
 changing, 5-18

compatible code pages, 6-5, 6-6

Continental Decimal Notation (CDN), C-16
 punctuating large numbers, C-16

CPCODEPG file, 5-3, 5-5

CPXCPTBL file, 5-3, 5-5, 5-6

currencies, C-2
 converting, C-2, C-3

CURRENCY attribute, C-8, C-9

currency data source, C-4
 activating, C-10
 creating, C-4, C-5, C-7
 CURRENCY attribute, C-8, C-9
 currency codes, C-4, C-6, C-7
 querying, C-11

D

data access, 6-2

data interpretation, 6-4

data source code pages, 6-9

date and time capabilities, 1-8

DBCS (Double-Byte Character Set), 1-3, 3-2, 3-11, G-4
 client and server configuration, 6-15
 code pages, 3-2, 3-7

decimals, G-4
 notation, 1-8

diacritics, G-4

DOS code page settings, 6-8

Double-Byte Character Set (DBCS), 1-3, 3-2, 3-11, G-4
 client and server configuration, 6-15
 code pages, 3-2, 3-7

E

EBCDIC (Extended Binary Coded Decimal Interchange Code), G-4

EBCDIK (Extended Binary Coded Decimal Interchange Kana Code), G-5

EDA, G-5
 default server code pages, 6-7

EDA Client, G-5

EDA Hub Server, 1-14, G-5
 combining data, 1-14

EDA Server, G-5

EDA Subserver, G-5

EUC (Extended UNIX Code), G-6

euro currency, C-2
 converting, C-2, C-3, C-12, C-14, C-15

EUROFILE parameter, C-10
 error messages, C-10, C-11
 restrictions, C-10

Extended Binary Coded Decimal Interchange Code (EBCDIC), G-4

Extended Binary Coded Decimal Interchange Kana Code (EBCDIK), G-5

extended characters, G-6

Extended UNIX Code (EUC), G-6

F

filters, G-6

fixed-width encoding, G-7

FOCSHELL, 6-7
 converting Master Files, 6-7
 using compatible code pages, 6-5

fonts, G-7

G

GCGID (Graphic Character Global Identifier), G-7

generation list files, 5-4, 5-5

Graphic Character Global Identifier (GCGID), G-7

graphic characters, G-7

H

hangul (Korean script), G-7

hanja (Korean ideographs), G-8

hexadecimal notation, G-8

hiragana Japanese script, G-8

HTML (Hypertext Markup Language), G-8

I

ideographs, G-8

International Organization for Standardization (ISO), 3-2, 3-7, 3-9, 3-11, G-9

internet, G-9

ISO (International Organization for Standardization), 3-2, 3-7, 3-9, 3-11, G-9

ISO code pages, 3-2, 3-7

J

Japanese Industrial Standard (JIS), G-9

JIS (Japanese Industrial Standard), G-9

joins, G-9

K

kana (Japanese scripts), G-9

kanji (Japanese ideographs), G-9

katakana (Japanese scripts), G-9

known code page files, 4-5, 4-6, 5-5, 5-6
alternate code pages, 5-5

known language list files, 4-9, 4-11, 5-9, 5-10, 5-11

L

LANGTBL file, 4-9, 4-11, 5-9, 5-10, 5-11

languages, 1-4, G-10

legacy OEM (DOS) code pages, 3-2, 3-7

local language error messages, 2-11, C-17

local language online help, 2-7

locale features, G-10

M

Master Files, 6-16

associating code pages, 6-16

monocasing, 4-12, 5-12, G-10

customizing for MVS, 5-12

customizing for Windows NT or UNIX, 4-12

monocasing tables

for MVS, 5-12

for Windows NT or UNIX, 4-12

multi-language code pages, 1-4

MVS server

adding additional MVS code pages, 5-8

adding alternate MVS code pages, 5-8

changing code page settings, 5-4

configuring for NLS, 5-2

identifying page settings, 5-4

NLS default characteristics, 5-9

MVS server code pages, 5-8

changing languages, 5-9

changing settings, 5-9

transcoding, 5-8

N

national characters, G-10

National Language Support (NLS), 1-1, 3-9, 6-1, G-11

- changing client code page settings, 4-17
- changing languages, 4-9, 4-10
- changing server code pages, 4-9, 4-10
- character mapping, 4-4
- configuring client and server, 6-13
- configuring for MVS, 5-1
- configuring for OpenEdition Client, 3-11
- configuring for UNIX Client, 3-9, 3-11
- configuring for UNIX Server, 3-2, 3-7
- configuring for WebFOCUS Windows, 6-1
- configuring for Windows NT Client, 3-9
- configuring for Windows NT Server, 3-2, 3-3
- custom configuring for Windows NT and UNIX, 4-1
- customizing monocasing tables, 4-12, 5-12
- customizing sort tables, 4-14
- identifying enabled language names, 4-9, 4-10, 5-9, 5-10
- initializing, 3-2
- installing, 6-9, 6-10
- monocasing, 1-2
- sorting, 1-2
- testing behavior, 4-17
- troubleshooting, 8-1, 8-2

National Language Versions (NLVs), 2-1, 2-2, 2-3, 2-4, 2-8, G-11

- changing local language, 2-12

NLS (National Language Support), 1-1, 3-9, 6-1, G-11

- changing client code page settings, 4-17
- changing languages, 4-9, 4-10
- changing server code pages, 4-9, 4-10
- character mapping, 4-4
- configuring, 3-9
- configuring client and server, 6-13
- configuring for MVS, 5-1
- configuring for OpenEdition Client, 3-11
- configuring for UNIX Client, 3-9, 3-11
- configuring for UNIX Server, 3-2, 3-7
- configuring for WebFOCUS Windows, 6-1
- configuring for Windows NT Client, 3-9
- configuring for Windows NT Server, 3-2, 3-3
- custom configuring for Windows NT and UNIX, 4-1

NLS (*continued*)

- customizing monocasing tables, 4-12, 5-12
- customizing sort tables, 4-14
- identifying enabled language names, 4-9, 4-10, 5-9, 5-10
- initializing, 3-2
- installing, 6-9, 6-10
- monocasing, 1-2
- sorting, 1-2
- testing behavior, 4-17
- troubleshooting, 8-1, 8-2

NLS API, 1-8, 1-9, G-11

- character mapping, 1-9
- date and time capabilities, 1-8
- decimal notation, 1-8
- monocasing, 1-8
- sorting, 1-8
- transcoding tables, 1-9

NLS configuration files, A-1, A-2, A-4

- for MVS, 5-2, 5-3
- for Windows NT or UNIX, 4-3

NLS Configuration Utility, 6-9

- local usage, 6-9

NLS sort tables, 5-14

nlscfg.err file, 4-17, 4-18, A-4

NLVs (National Language Versions), 2-1, 2-2, 2-3, 2-4, 2-8, G-11

- changing local language, 2-12

O

odin.cfg file, 4-17, 4-18

OEM (Original Equipment Manufacturer), G-11

Original Equipment Manufacturer (OEM), G-11

P

procedure code pages, 6-9

- R**
- radix characters, G-12
 - romaji (Latin script), G-12
- S**
- SBCS (Single-Byte Character Set), 1-3, G-12
 - SBCS (Single-Byte Character Sets)
 - configuring client and server, 6-14, 6-15
 - scripts, 1-4, G-12
 - server code pages, 6-7
 - changing languages, 5-9
 - EDA default, 6-7
 - server JCL, 5-17
 - servers, 3-2, G-12
 - components, 3-2, 3-11
 - translation, 3-2, 3-11
 - Shift-JIS encoding, G-12
 - Simplified Chinese characters, G-13
 - Single-Byte Character Set (SBCS), G-12
 - Single-Byte Character Sets (SBCS), 1-3
 - configuring client and server, 6-14, 6-15
 - sort sequences, 4-14
 - sorting tables
 - for MVS, 5-14
 - for Windows NT or UNIX, 4-14
 - sorttbl.err file, A-2
- T**
- thousands separator symbol, G-13
 - transcoding, 1-3, 4-8, 5-8, A-2, G-13
 - generating tables, A-2
 - Transcoding Services Generation Utility (TSGU), 4-8, 5-8, A-1, A-2
 - info-commands, A-2, A-3
 - troubleshooting for NLS, 8-1, 8-2, 8-3
 - correcting installation, 8-9
 - determining code pages, 8-6, 8-7
 - identifying NLS problems, 8-4
 - improper configuration, 8-5
 - Information Builders support, 8-3, 8-11
 - solving display issues, 8-10
 - solving printing issues, 8-10
 - TSGU (Transcoding Services Generation Utility), 1-9, 4-8, 5-8, A-1, A-2
 - info-commands, A-2, A-3
 - TSGU info-commands, A-2
- U**
- UCS (Universal Character Set), G-14
 - Unicode, 1-16, 7-1, 7-2, 7-3, 7-5, B-4, G-14
 - combining scripts, 1-16, 7-5
 - configuring systems, 7-3, 7-4
 - Unicode Transformation Format (UTF), G-14
 - Universal Character Set (UCS), G-14
 - UNIX, 3-1, 4-1
 - changing code page settings, 4-4
 - customizing for NLS, 4-1
 - user interface, 6-2
 - UTF (Unicode Transformation Format), G-14
 - UTF-16 encoding form, 7-2, G-14
 - UTF-8 encoding form, 7-2, G-14
- W**
- WebFOCUS (Windows version)
 - data access, 6-2
 - user interface, 6-2
 - Windows code page settings, 6-8
 - Windows NT, 3-1, 4-1
 - customizing for NLS, 4-1
 - Windows NT server code pages, 4-4
 - changing settings, 4-4

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services – Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: <http://www.informationbuilders.com/bookstore/derf.html>

Name: _____

Company: _____

Address: _____

Telephone: _____ Date: _____

E-mail: _____

Comments:

Reader Comments