

WebFOCUS

National Language Support
Release 8.2 Version 02

Active Technologies, EDA, EDA/SQL, FIDEL, FOCUS, Information Builders, the Information Builders logo, iWay, iWay Software, Parlay, PC/FOCUS, RStat, Table Talk, Web390, WebFOCUS, WebFOCUS Active Technologies, and WebFOCUS Magnify are registered trademarks, and DataMigrator and Hyperstage are trademarks of Information Builders, Inc.

Adobe, the Adobe logo, Acrobat, Adobe Reader, Flash, Adobe Flash Builder, Flex, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. 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 © 2017, by Information Builders, Inc. and iWay Software. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Contents

Preface	9
Documentation Conventions	10
Related Publications	12
Customer Support	12
Information You Should Have	12
User Feedback	13
Information Builders Consulting and Training	14
1. Understanding Globalization and the NLS API	15
What Is National Language Support?	16
National Language Support and Localized Versions	16
Understanding Code Pages	17
Character Sets.	17
Defining Scripts, Languages, and Code Pages.	17
Code Point Representation of Written Symbols	20
Code Pages in Web-based Computing	20
Information Builders NLS API	21
How NLS API Works	22
Multiple Client Code Pages on Different Platforms	24
Reading Data in Different European Languages With a Single Code Page	24
Combining Data From Different Code Pages With Subservers	26
What is Unicode?	27
Why Use Unicode?.....	28
Determining Whether Unicode Is Necessary.....	28
Defining the Alphanumeric Data Type.....	29
Combining Data in Unrelated Scripts With Unicode.....	30
2. Working With Localized Versions	33
What Is a Localized Version?	33
NLV Excel Add-In for Quick Data.....	34
NLS and NLV of WebFOCUS Open Portal Services.....	34
Localized Version Components.....	34
Installing a Localized Version	34

WebFOCUS Reporting Server.....	35
WebFOCUS Client and ReportCaster Distribution Server.....	35
WebFOCUS App Studio.....	35
Using the Dynamic Language Switch	36
Switching Between Languages.....	39
Passing the Language Value to the URL.....	42
Creating a Localized Version	44
Translating a Properties File.....	47
Translating a JavaScript Resource File.....	49
Translating a Text File.....	50
Using a Localized Version	52
Accessing Local Language Online Help	57
Working With an Unfamiliar Local Language	57
Interpreting Local Language Error Messages	57
3. Configuring the WebFOCUS Reporting Server for NLS	59
Configuring a WebFOCUS Reporting Server for NLS	59
Determining Which Code Page You Need.....	68
Localizing the WebFOCUS Reporting Server Console	71
Configuring National Language Support for the Data Management Console	78
4. Configuring a WebFOCUS Client for NLS	79
Configuring a WebFOCUS Client for NLS	79
Displaying National Characters on Sun Solaris	81
Adding NLS Information to the Communications Configuration File	82
ReportCaster Distribution Server Support for UTF-8 on Windows	83
5. Configuring WebFOCUS App Studio for NLS	87
App Studio Architecture	87
App Studio Configuration Steps	88
Step 1. Verify That the Server Is Configured for the Code Page of the Data Source.....	88
Step 2. Verify That the Client is Configured for the Correct Display of Report Output.....	88
Step 3. Configure App Studio for NLS.....	88
App Studio Configuration Summary	89
Configuring for Single-Byte Character Languages Accessing ANSI (Windows) Data.....	89

Configuring for Double-Byte Character Languages.....	90
6. Unicode and the WebFOCUS Reporting Server	91
Unicode Support on the WebFOCUS Reporting Server	91
Accessing Unicode Data	93
Selecting, Reformatting, and Manipulating Characters	100
Sort Order Under Unicode	103
Create a Unicode Environment Without Using the Tomcat Administration Tool	103
Unicode PDF Output	104
7. Troubleshooting	105
Troubleshooting Strategy	105
Questions to Ask.....	106
Observing Results.....	107
Additional Resources.....	107
Identifying NLS Issues	107
What Should You Do?.....	108
Determining NLS Configuration Values	109
Examining the Code Page Generation File	116
Changing NLS Configuration Settings	116
8. Display, Print, and Language Considerations	117
Display Issues	117
Print Issues	118
WebFOCUS Language Considerations	119
A. History of Code Pages	121
Origin of Code Pages	121
Far Eastern Encodings	122
Unicode	123
Code Page Families	123
B. Helpful NLS Commands	125
Setting the Language of Excel in the NLS Configuration File	126
Setting the Excel Version Number	127
Setting Column Width for Excel in the NLS Configuration File	128
Punctuating Large Numbers	129

Displaying a Leading Zero in Decimal-only Numbers	131
Selecting an Extended Currency Symbol	132
Using a Currency ISO Code.....	138
Setting Date and Time Display and Formatting	139
DATE_ORDER.....	139
DATE_SEPARATOR.....	140
TIME_SEPARATOR.....	141
Setting Business Days and Holidays	141
BUSDAYS.....	141
HDAY.....	141
WEEKFIRST.....	142
Determining Collation of Data	143
COLLATION.....	143
Displaying National Characters in Server-Side Graphics on UNIX and Linux	144
Specifying Multilingual Metadata in a Master File	146
Storing Localized Metadata in Language Files	153
LNGPREP Utility: Preparing Metadata Language Files.....	153
LNGPREP Modes.....	155
Using LNGPREP for a Multi-Locale Application.....	158
Adding the Code Page Parameter to a Master File	158
Running PDF or PS Reports with International Fonts in App Studio	160
C. Helpful NLS Functions	163
BYTVAL: Translating a Character to Decimal	164
CHAR: Returning a Character Based on a Numeric Code	166
CTRAN: Translating One Character to Another	167
DCTRAN: Translating A Single-Byte or Double-Byte Character to Another	169
DATECVT: Converting the Format of a Date	170
DATETRAN: Formatting Dates in International Formats	172
HDATE: Converting the Date Portion of a Date-Time Value to a Date Format	188
HEXBYT: Converting a Decimal Integer to a Character	189
HEXTYPE: Returning the Hexadecimal View of an Input Value	191
JPTRANS: Converting Japanese Specific Characters	192

KKFCUT: Truncating a String	197
LCWORD: Converting a String to Mixed-Case	198
LCWORD2: Converting a String to Mixed-Case	200
LCWORD3: Converting a String to Mixed-Case	201
LOCASE: Converting Text to Lowercase	202
PATTERN: Generating a Pattern From a String	203
REPLACE: Replacing a String	205
REVERSE: Reversing the Characters in a String	207
STRIP: Removing a Character From a String	209
DSTRIP: Removing a Single-Byte or Double-Byte Character From a String	211
SFTDEL: Deleting the Shift Code From DBCS Data	212
SFTINS: Inserting the Shift Code Into DBCS Data	213
STRREP: Replacing Character Strings	215
UFMT: Converting an Alphanumeric String to Hexadecimal	217
UPCASE: Converting Text to Uppercase	218
D. Creating Multi-Locale Applications	221
Creating a Multi-Locale Application Using LNGPREP	221
Displaying Multilingual Reports	230
How Does the Sample Application Work?.....	231
Development Steps.....	233
Localizing Portals	239

Preface

This content provides information on National Language Support (NLS) for WebFOCUS. It contains instructions for configuring the WebFOCUS Reporting Server, WebFOCUS Client, and ReportCaster Distribution Server for NLS on Windows®, UNIX®, Linux®, IBM® i, and z/OS®. It also provides an introduction to software internationalization, which is the process of making software operable in multiple locales.

This manual is intended for the person responsible for planning the enterprise's NLS environment and for configuring WebFOCUS components for NLS. It is also helpful to anyone who wants to become more familiar with NLS concepts and architecture in particular, or internationalization in general.

How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	Understanding Globalization and the NLS API	Defines software internationalization and globalization, and introduces NLS concepts and features, including Unicode®, a universal character set that includes every written symbol in all the living languages of the world. Describes code page architecture and the implementation of NLS at Information Builders. Also contains scenarios and diagrams for possible NLS configurations.
2	Working With Localized Versions	Provides installation and usage information for localized versions, which are software products that display the user interface in a local language other than English. Explains how to dynamically switch between languages used for the interface and how to customize the interface.
3	Configuring the WebFOCUS Reporting Server for NLS	Describes the steps for configuring a WebFOCUS Reporting Server for NLS on Windows, UNIX, IBM i, or z/OS, using the supplied web-based consoles.
4	Configuring a WebFOCUS Client for NLS	Describes the steps for configuring a WebFOCUS Client for NLS on Windows, UNIX, IBM i, or z/OS, using the supplied web-based consoles.

Chapter/Appendix		Contents
5	Configuring WebFOCUS App Studio for NLS	Describes the architecture of WebFOCUS App Studio and the configuration of App Studio for NLS. App Studio is a Windows-based environment for creating and deploying web-based reporting applications. It provides intuitive GUI tools for building a user interface.
6	Unicode and the WebFOCUS Reporting Server	Describes WebFOCUS Reporting Server support for Unicode.
7	Troubleshooting	Discusses techniques and resources for identifying NLS problems and determining solutions. Describes how to use the WebFOCUS Reporting Server Console to display NLS configuration information that will help you in problem determination.
8	Display, Print, and Language Considerations	Describes display, printing, and WebFOCUS Language considerations for NLS.
A	History of Code Pages	Discusses the evolution of code pages and their standardization by a number of organizations.
B	Helpful NLS Commands	Gives the syntax of commands that optimize your implementation of NLS and includes procedures for generating and displaying national characters.
C	Helpful NLS Functions	Gives the syntax of functions that can be used to localize your data for use in procedures, or aid in converting your data between formats and code pages.
D	Creating Multi-Locale Applications	Describes ways to structure multi-locale, self-service applications that display content in multiple languages from a single file.

Documentation Conventions

The following table lists and describes the conventions that apply in this manual.

Convention	Description
<p><code>THIS TYPEFACE</code></p> <p>or</p> <p><code>this typeface</code></p>	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), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
this typeface	Highlights a file name or command.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices; type one of them, not the braces.
[]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

To view a current listing of our publications and to place an order, visit our Technical Content Library, <http://documentation.informationbuilders.com>. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about this product?

Call Information Builders Customer Support Services (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 questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Be prepared to provide your six-digit site code (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 website, <https://techsupport.informationbuilders.com>. You can connect 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 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 effectively, be prepared to provide the following information when you call:

- ☐ Your six-digit site code (xxxx.xx).
- ☐ Your WebFOCUS configuration:
 - ☐ The front-end software you are using, including vendor and release.
 - ☐ The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
 - ☐ The software release.
 - ☐ Your server version and release. You can find this information using the Version option in the Web Console.

- ☐ Your NLS configuration, including the code page for the server, client, repository database, and operating system.
- ☐ The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- ☐ 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 just the code to access the 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?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff welcomes your opinions regarding this manual. Please use the Reader Comments form at the end of this manual to communicate suggestions for improving this publication or to alert us to corrections. You can also use the Documentation Feedback form on our website, <http://documentation.informationbuilders.com/feedback.asp>.

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 website (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

Understanding Globalization and the 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 the seamless interpretation of all national characters embedded in any data source.

A proprietary NLS API code page engine enables Information Builders products to work in all countries and for all languages. The main function of the NLS API is to manage the transcoding of data between client and server components, each of which may have different code pages. The NLS API also provides sorting, case conversion, and the formatting of dates, time, currency, and numbers based on the conventions of a locale.

Information Builders NLS API supports Unicode, a universal character encoding standard that assigns a code to every character and symbol in every language in the world. Unicode is the only encoding standard that ensures that you can retrieve or combine data using any combination of languages. The WebFOCUS Reporting Server supports a Unicode Transformation Format (UTF) called UTF-8 in ASCII environments and UTF-EBCDIC in EBCDIC environments.

In this chapter:

- ☐ [What Is National Language Support?](#)
- ☐ [National Language Support and Localized Versions](#)
- ☐ [Understanding Code Pages](#)
- ☐ [Code Point Representation of Written Symbols](#)
- ☐ [Code Pages in Web-based Computing](#)
- ☐ [Information Builders NLS API](#)
- ☐ [How NLS API Works](#)
- ☐ [Multiple Client Code Pages on Different Platforms](#)
- ☐ [Reading Data in Different European Languages With a Single Code Page](#)
- ☐ [Combining Data From Different Code Pages With Subservers](#)

☐ What is Unicode?

What Is National Language Support?

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 support involves transcoding each graphic character on one code page into the corresponding graphic character on another code page.

For example, to transcode the German graphic character Ä from IBM mainframe EBCDIC German code page 273 to its corresponding value in Windows Western European code page 1252, the Information Builders transcoding engine changes hexadecimal value 0x4A (mainframe) to hexadecimal value 0xC4 (Windows). For more information on code pages, see [Understanding Code Pages](#) on page 17.

Additional NLS functions include:

- ☐ **Sorting.** A sort order is defined for each code page.
- ☐ **Monocasing.** Monocasing (also called case conversion) converts a letter from its lowercase to uppercase form (or vice versa).
- ☐ **Date and time formats.** You can select date and time formats for local languages.
- ☐ **Decimal notation and large number formatting.** You can select proper decimal notation and formatting for large numbers and currency for local languages.

National Language Support and Localized Versions

National Language Support (NLS) and localized versions are frequently confused. NLS ensures that systems can handle local language data. A localized version is a software product in which the entire user interface appears in a particular language. The user interface includes:

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

Examples of Information Builders localized versions are the Brazilian Portuguese, Chinese, French, German, Italian, Japanese, and Spanish editions of WebFOCUS. Information Builders localized versions include both the translated and localized user interface, and handle local language data as well. For information on installing and enabling a localized version, switching between languages, customizing the user interface, and working with the features of a localized version, see [Working With Localized Versions](#) on page 33.

Understanding Code Pages

A code page is a matching set of numeric values and the written symbols they represent. It is usually defined by the vendor of an operating system (platform). The process of transcoding data between the client and server is the actual mapping of the graphic characters on one code page to the corresponding graphic characters on another code page.

Code pages have the following characteristics:

- ❑ They support a specific language and a specific platform.
- ❑ On 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. For more information on the code page families used by Information Builders products, see [Code Page Families](#) on page 123.
- ❑ 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:

- ❑ **Single-Byte Character Set (SBCS)** code pages are 8-bit encodings that 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 and Japanese) 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 topic will give you a better understanding of the differences in these terms.

A script is a collection of symbols that 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 major writing systems of the world, 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. The set of English characters is generally referred to as the ASCII character set.

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), and French accented characters (á, â, ã).

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 code page 1252, which handles all major North American, South American, and Western European languages (except Greek) for Windows and UNIX. Code page 1252 is functionally equivalent to Microsoft Windows 1252 and UNIX ISO 8859-1 code pages.

***Reference:* Information Builders Key Reporting Server Code Pages**

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

Language	Windows	UNIX Linux	z/OS (PDS and HFS Deployments) IBM i
English	1252 (default)	1252 (default)	37 (default)
Western European	1252	1252	37 or a code page dependent on country
Central European (Polish and Czech)	1250	1250	870
Turkish	1254	1254	1026
Lithuanian	1257	1257	1112
Latvia	1257	1257	1112
Estonian	1257	1257	1112
Traditional Chinese	10948	10948	937
Japanese	942	10942/942	939/930
Hebrew	1255	1255	424
Unicode	65001	65001	65002

Note:

- ☐ OS/390, z/OS, MVS, VM, and IBM i all use an IBM operating system and share the same family of EBCDIC code pages.
- ☐ VM applies to an iWay Subserver for VM connected to a WebFOCUS Reporting Server.
- ☐ OpenVMS applies to an iWay Subserver for OpenVMS connected to a WebFOCUS Reporting Server.

Code Point Representation of Written Symbols

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

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

Windows 1252 code page

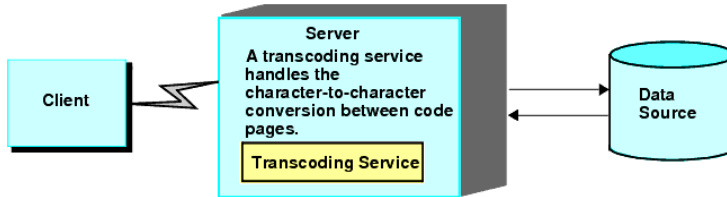
	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
Code point 0	0	@	P	`	p	€	□	NSP	°	À	Đ	à	ó		
Code point 0x41	1	!	1	(A)	Q	a	q	□	´	ı	±	Â	Ñ	á	ñ
	2	"	2	B	R	b	r	,	´	ç	²	Ã	Ò	â	ò
	3	#	3	C	S	c	s	f	"	£	³	Ä	Ó	ã	ó
	4	\$	4	D	T	d	t	„	"	¤	´	Å	Ô	ä	ô
	5	%	5	E	U	e	u	...	•	¥	µ	Ä	Õ	å	õ
	6	&	6	F	V	f	v	†	—	ı	¶	Æ	Ö	æ	ö
	7	´	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
	8	(8	H	X	h	x	^	*	™	„	È	Ø	è	ø
	9)	9	I	Y	i	y	%	™	©	ˆ	É	Ù	é	ù
	A	*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
	B	+	;	K	[k	{	<	>	«	»	Ë	Û	ë	û
Unassigned code point	C	,	<	L	\	ı		Œ	œ	¬	¼	İ	Ü	ı	ü
	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 Web-based Computing

Internet or intranet products, such as WebFOCUS, often have their components installed on different computers. These computers can have different operating systems. Since code pages are platform-specific, components 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 the letter A on the client code page.

The following diagram illustrates the generic code page architecture for web-based computing.



In this topic, *server* refers to a WebFOCUS Reporting Server. *Client* can refer to any of the following components that connect to a Reporting Server:

- ☐ The WebFOCUS Common Gateway Interface (CGI), ISAPI, or Java servlets.
- ☐ App Studio.
- ☐ A third-party product.

Information Builders NLS API

The National Language Support (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 built into WebFOCUS and most other Information Builders products.

The graphic character representation standards (including code page numbers, layouts, and binary values for specific written symbols) that are used by Information Builders comply with IBM's Character Data Representation Architecture (CDRA).

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

- ☐ **Sorting.** Creates rules for a sort order in a language.
- ☐ **Monocasing.** Also called case conversion, monocasing 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 E.

The NLS API provides standard sorting and monocasing tables for supported code pages. To customize these tables for your enterprise, see your Information Builders representative.

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 [Punctuating Large Numbers](#) on page 129.
- ❑ **Date and time capabilities.** Since there are numerous date and time formats used throughout the world, the NLS API enables you to select 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 the syntax of available date and time formats, see the *Describing an Individual Field* chapter in the *Describing Data With WebFOCUS Language* manual.

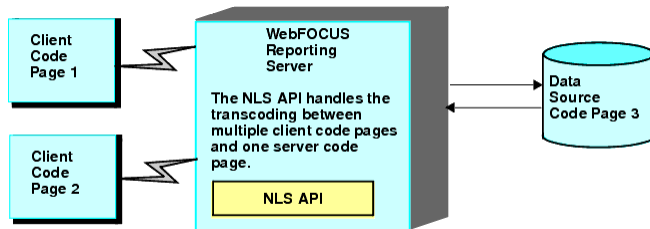
How NLS API Works

The NLS API code page architecture is Information Builders proprietary implementation of the standard way of handling code page conversions used by all heterogeneous multi-tier systems. A server can have one server code page for reading data sources but can support multiple client code pages.

During the WebFOCUS configuration process, the administrator selects a code page for the WebFOCUS Reporting Server and WebFOCUS Client. For details on configuring components for NLS, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59 and [Configuring a WebFOCUS Client for NLS](#) on page 79.

During configuration, 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) generates the code page transcoding tables.

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



To view these, see:

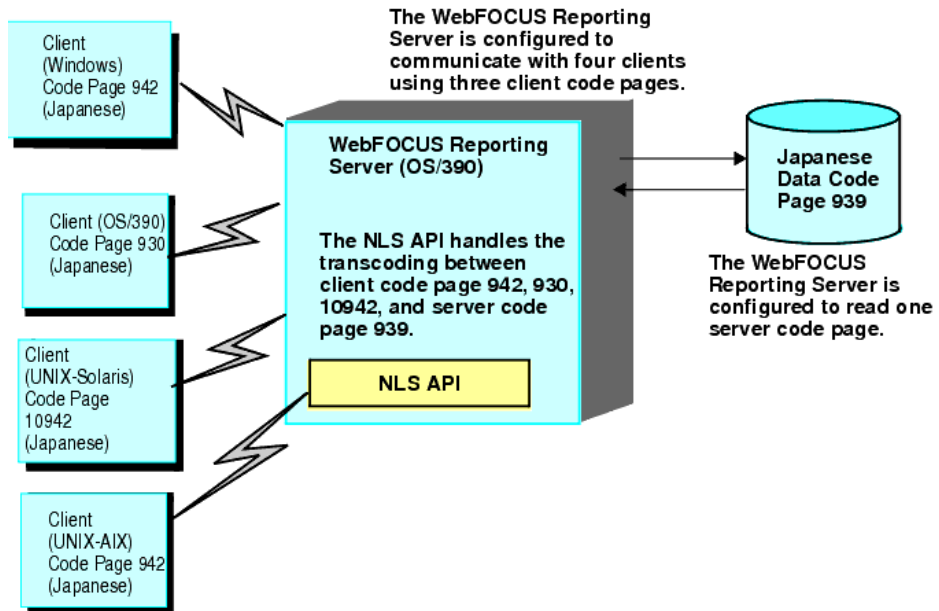
- ❑ [Multiple Client Code Pages on Different Platforms](#) on page 24.
- ❑ [Reading Data in Different European Languages With a Single Code Page](#) on page 24.
- ❑ [Combining Data From Different Code Pages With Subservers](#) on page 26.
- ❑ [Combining Data in Unrelated Scripts With Unicode](#) on page 30.

Multiple Client Code Pages on Different Platforms

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

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 WebFOCUS Reporting Server is configured to communicate with three client code pages. All Reporting Servers can support multiple client code pages.
2. The WebFOCUS Reporting Server is configured to read data with one server code page. All Reporting 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.



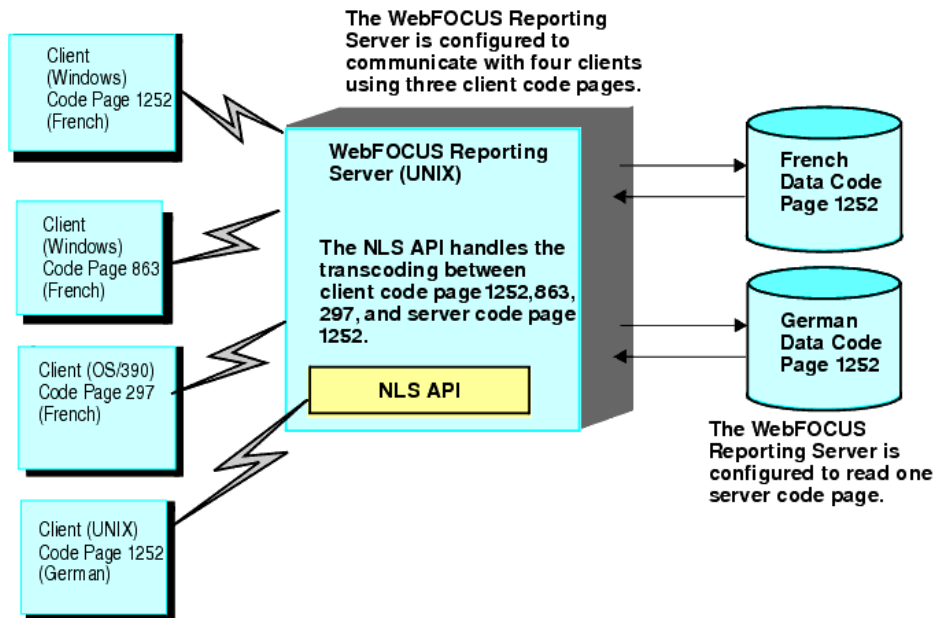
Reading Data in Different European Languages With a Single Code Page

In the following scenario, the two data sources are in different languages (French and German), but can be read with the same server code page (1252).

Information Builders code page 1252 is based on the Latin script and contains all symbols for the languages of North America, South America, and Western Europe. It is implemented on both the Windows and UNIX platforms. For related information on the Latin script, see [Defining Scripts, Languages, and Code Pages](#) on page 17.

1. Both data sources use code page 1252. If the two data sources have a common index key with a common range of values, you can join them for reporting. Otherwise, you can make separate requests against each data source.
2. The WebFOCUS Reporting Server is configured to communicate with three client code pages. All Reporting Servers can support multiple client code pages.
3. A request from a client is passed through the NLS API on the WebFOCUS Reporting Server 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. This solution works only when data sources use the same code page. In this scenario, code page 1252 handles French and German characters and is available on both the Windows and UNIX platforms. If data sources require different code pages, subervers are needed to combine them.

For details on combining different code pages with subervers, see [Combining Data From Different Code Pages With Subervers](#) on page 26.

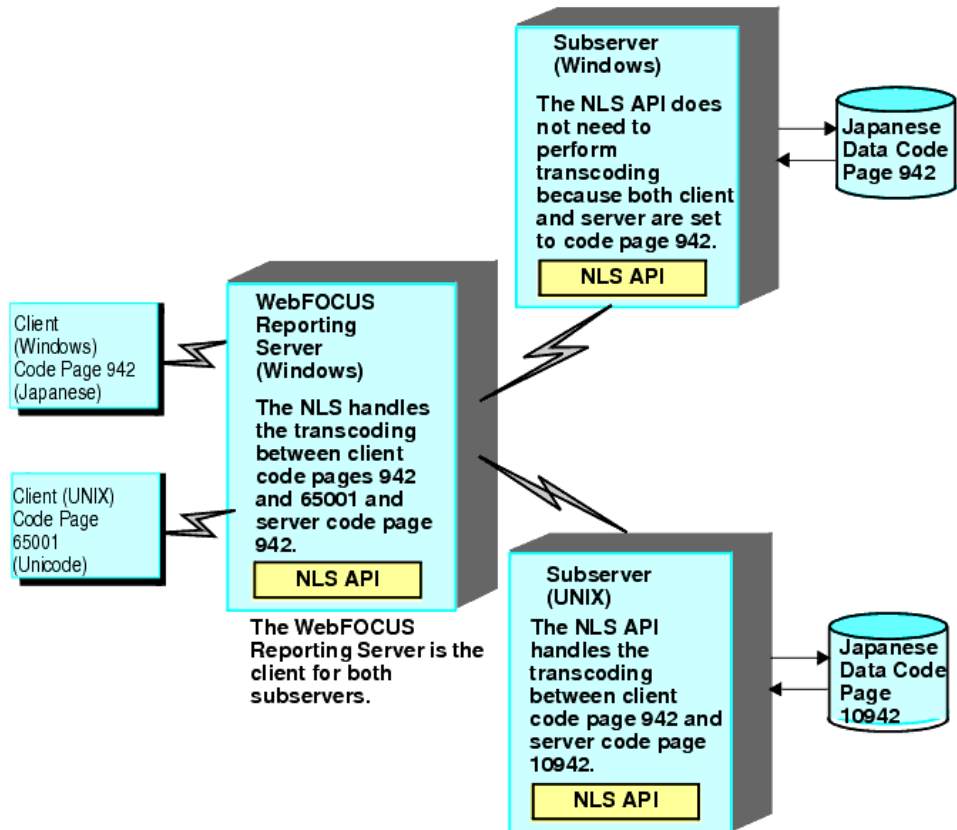


Combining Data From Different Code Pages With Subservers

Data sources using different code pages for the same or different languages can be joined using a WebFOCUS Reporting Server along with subservers. The following describes 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.
 - ❑ Shift-JIS is PC data. It resides on a Windows subserver.
 - ❑ EUC is UNIX data. It resides on a UNIX subserver.
2. While the two subservers have different code page settings, both are configured to communicate with the same client code page (Shift-JIS code page 942). The WebFOCUS Reporting Server is the client for both subservers. Therefore, the server code page setting for the WebFOCUS Reporting Server (Shift-JIS code page 942) is the same as the client code page setting for the subservers (Shift-JIS code page 942).
3. The WebFOCUS Reporting Server is configured to communicate with two client code pages (Shift-JIS code page 942 and UTF-8 Unicode code page 65001).
4. The Windows client could be a browser that is configured to read Shift-JIS, so it reads Japanese data from code page 942. The UNIX client could be a WebFOCUS Client 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.



Note: You can also combine unrelated scripts using a Unicode solution. For more information, see [What is Unicode?](#) on page 27 and [Combining Data in Unrelated Scripts With Unicode](#) on page 30.

What is Unicode?

Unicode is a universal character encoding standard that assigns a code to every character and symbol in every language in the world. Since no other encoding standard supports all languages, Unicode is the only encoding standard that ensures that you can retrieve or combine data using any combination of languages. Unicode is required with XML, Java, JavaScript, LDAP, and other web-based technologies.

The two common Unicode implementations for computer systems are UTF-8, a variable length encoding scheme in which each written symbol is represented by a one- to four-byte code, and UTF-16, a fixed width encoding scheme in which each written symbol is represented by a two-byte code.

Information Builders supports UTF-8 Unicode on the WebFOCUS Reporting Server. The introduction of Unicode into Information Builders core text-handling facilities gives its products the flexibility to support true multilingual text. Using Unicode formatted data, WebFOCUS can produce a report containing data in any combination of languages (for example, Japanese and French). For details, see [*Unicode and the WebFOCUS Reporting Server*](#) on page 91.

UTF-8 Unicode is not a code page, but it is treated as such in Information Builders product architecture. Information Builders has created a code page of UTF-8 values for all supported scripts.

Why Use Unicode?

Unicode supports data with multiple scripts such as French, Japanese, and Hebrew. It enables you to combine records from different scripts on a single report. Before Unicode, a computer could only process and display the written symbols on its operating system code page, which was tied to a single script. For example, if a computer could process French, it could not process 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®, and many others. Many of the important data sources WebFOCUS accesses now support Unicode data types. The introduction of Unicode into WebFOCUS 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 Google Chrome and Firefox. Unicode is also used internally in Java technologies, HTML, XML, and Windows and Office. 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.

Determining Whether Unicode Is Necessary

Configure your system for Unicode if you need to display text in unrelated scripts. There may be situations in which Unicode appears to be the only way to assimilate scripts, because you need to include third-party Unicode data. However, in many cases, Unicode is not the only solution.

For example, if you have Oracle data with a UTF-8 Unicode data type, but all the text is in Japanese and English, you do not need a full Unicode implementation. Japanese and English are not unrelated scripts. A Japanese code page is ASCII transparent. An ASCII-transparent code page contains the standard English language characters, as well as additional non-English characters.

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

Important: Full use of Unicode requires careful attention to browser, web server, and operating system characteristics. These characteristics include international language fonts, display and print features, and data input for unfamiliar scripts such as Chinese, and Japanese. Implement Unicode only if there is a real business need to combine unrelated scripts.

If you confirm that data in a data source (for example, Oracle) is coded in Unicode format, with text in different languages (for example, Japanese and English), you have two configuration options:

- ❑ Configure the WebFOCUS Reporting Server code page for UTF-8 Unicode to access the Oracle data, but set the server client code page for standard Japanese/English, such as Shift-JIS for Windows. All the Japanese and English text appears correctly in a browser set to the Shift-JIS encoding.

or

- ❑ Change the Oracle Relational Database Management System client code page to Shift-JIS, and configure the WebFOCUS Reporting 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 the relational and non-relational database management systems supported by Information Builders have the same code page architecture as WebFOCUS, 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 documentation of the specific vendor.

Defining the Alphanumeric Data Type

UTF-8 is a variable-length encoding scheme. In general, 7-bit ASCII characters (familiar English letters) are one byte, many European extended (national) characters are two bytes, and Double-Byte Character Set symbols (Japanese kanji) are three bytes.

Familiar alphanumeric data definitions such as SQL CHAR (*n*) or WebFOCUS 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.

Combining Data in Unrelated Scripts With Unicode

In the following scenario, Unicode is the only solution for combining data in two unrelated scripts, such as Japanese and Chinese. Without Unicode implementation, Japanese and Chinese, as unrelated scripts, are accessed using different code pages regardless of operating system. See the diagram in [Combining Data in Unrelated Scripts With Unicode](#) on page 30 to see how unrelated scripts are combined.

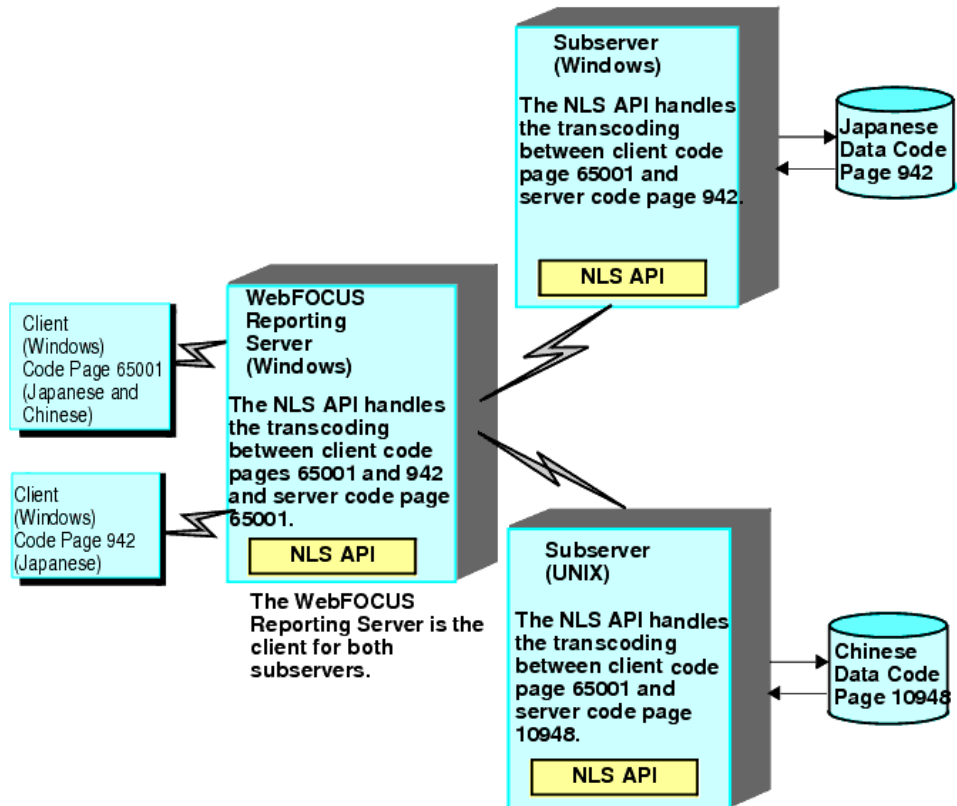
In this scenario, the data sources may or may not be on two different operating systems.

Example: Combining Data in Unrelated Scripts With Unicode

This example shows how to combine data in unrelated scripts with Unicode.

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 code page 10948.
 - ❑ Shift-JIS is Japanese PC data. It resides on a Windows subserver.
 - ❑ Big-5 is Traditional Chinese UNIX or PC data. It can reside on either a UNIX platform or PC workstation. In this example, it resides on a UNIX subserver.
2. While the two subservers have different server code page settings, both are configured to communicate with the same client code page (UTF-8 Unicode code page 65001). A UTF-8 Unicode configuration is the only solution for combining unrelated scripts.
3. The WebFOCUS Reporting Server is the client for both subservers. Therefore, the subserver code page setting on the WebFOCUS Reporting Subserver (UTF-8 Unicode code page 65001) is the same as the client code page setting for the subservers (UTF-8 Unicode code page 65001).
4. The WebFOCUS Reporting Server is configured to communicate with two client code pages (UTF-8 Unicode code page 65001 and Shift-JIS code page 942).
5. One Windows client is a browser that is set to UTF-8 Unicode, so it can read both Japanese and Chinese data from code page 65001. The other Windows client is a browser that is set to Shift-JIS, so it can read Japanese data from code page 942. However, any Chinese data this second browser encounters will be displayed incorrectly.

6. Multiple WebFOCUS Reporting Servers can usually run on the same computer as separate, intercommunicating processes. Therefore, all data sources, servers, subservers, and browsers could potentially run on a single machine. Shift-JIS (Japanese) and Traditional Chinese are unrelated scripts and cannot be combined without a Unicode solution. The processes may also run on separate computers, including a combination of UNIX and PC workstations as shown in the following image.



Working With Localized Versions

A localized version is a software product that displays the user interface in a local language such as French, German, or Spanish. Information Builders provides localized versions in selected languages for business intelligence products, such as WebFOCUS.

In this chapter:

- ☐ [What Is a Localized Version?](#)
 - ☐ [Installing a Localized Version](#)
 - ☐ [Using the Dynamic Language Switch](#)
 - ☐ [Creating a Localized Version](#)
 - ☐ [Using a Localized Version](#)
 - ☐ [Accessing Local Language Online Help](#)
 - ☐ [Working With an Unfamiliar Local Language](#)
 - ☐ [Interpreting Local Language Error Messages](#)
-

What Is a Localized Version?

A localized version is a software product in which the entire user interface (for example, menus, utilities, and online Help) appears in a particular language.

Information Builders localized versions are positional. Each button or menu choice is always in the same relative position in the product. Only the text of the labels changes. Therefore, it is easy for anyone who is familiar with an Information Builders product in one language to support that product in another language.

Examples of Information Builders localized versions are the Chinese (Simplified and Traditional), French (Canadian), French (Standard), German (Standard), Italian, Japanese, Portuguese (Brazilian), and Spanish.

Localized versions have complete National Language Support (NLS) capability. You can access data for any supported language, with all national characters processed, displayed, and printed correctly.

NLV Excel Add-In for Quick Data

WebFOCUS Quick Data is a Microsoft Office Add-in that enables you to connect Microsoft Excel directly to WebFOCUS reporting tools, where you can access and analyze all of your enterprise data. Connecting Excel to WebFOCUS allows WebFOCUS Quick Data to leverage over 300 adapters to provide you with access to practically an unlimited amount of enterprise information.

A localized version provides a the WebFOCUS Quick Data plug-in to Microsoft Excel. The plug-in for Excel is included with App Studio, where it is added to the Excel plug-in folder and implemented through a WebFOCUS button on the Excel menu. When you launch Quick Data from a localized version of Excel, the dialog boxes are localized in the following languages: French, German, Spanish, Japanese, Traditional Chinese, and Simplified Chinese.

NLS and NLV of WebFOCUS Open Portal Services

National Language Support (NLS) and National Language Version (NLV) support is available for WebFOCUS Open Portal Services. All Java Portlet Specification 2.0 (JSR 286-compliant) portal environments are supported (for example, Microsoft SharePoint and IBM WebSphere Portal Server). For more information on configuring and using WebFOCUS Open Portal Services, see the *WebFOCUS Embedded Business Intelligence User's Guide*.

Localized Version Components

The following user interface components are translated in a localized version:

- ☐ Initial program installation
- ☐ Menus, toolbars, dialog boxes, forms
- ☐ Program utilities (assistants, wizards, editors)
- ☐ Error messages
- ☐ Online Help, including context-sensitive Help, which is added to each version of WebFOCUS in a service pack

Installing a Localized Version

An installation program in a supported local language is provided for the WebFOCUS Reporting Server, the WebFOCUS Client, WebFOCUS App Studio, and other Information Builders software. The sequence of steps in the installation procedure is identical to the English version of the product.

By default, the installation program uses the same language specified in the operating system's regional settings or browser language options. If that language is not available in a localized version, English is used instead.

WebFOCUS Reporting Server

With the exception of the sample application files used for demos, all WebFOCUS Reporting Server localized versions are included with every installation. To install localized demos, select from the list of available language options when prompted during the server installation procedure. For an illustration, see [Language Selection for Demo Files](#) on page 35.

For a full description of the installation procedure, see the *WebFOCUS and ReportCaster Installation and Configuration* manual for your operating system.

In addition, you can localize the Reporting Server Web Console for languages that are not automatically included with the installation. Since this procedure is closely related to the server configuration for NLS, instructions for localizing the Web Console are described in detail in [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59.

Example: Language Selection for Demo Files

During WebFOCUS Reporting Server installation, select the language for the New Century Corporation demo application.

`New Century Corporation Demo files. Select languages to install (be sure to select corresponding demo files during WebFOCUS installation).`

- `1. English`
- `2. French`
- `3. German`
- `4. Spanish`

`Type choices separated by spaces or press Enter to bypass demo installation.`

The demo file column titles will match the language set for the Reporting Server if a translated version of that file is available. For example, if the Reporting Server is running in Italian, the wf_retail demo uses Italian column titles.

WebFOCUS Client and ReportCaster Distribution Server

A single WebFOCUS Client installation supports all localized versions.

WebFOCUS App Studio

A single WebFOCUS App Studio installation supports all localized versions.

Using the Dynamic Language Switch

The dynamic language switch enables users to dynamically toggle between user interface languages from their sign-in pages without affecting other users. A WebFOCUS administrator uses the WebFOCUS Administration Console to enable the dynamic language switch and to control the languages available for switching.

Procedure: How to Enable the Dynamic Language Switch in WebFOCUS

- 1. Sign in to WebFOCUS with administrative privileges.
- 2. In the Administration menu, click *Administration Console*.

The Administration Console opens to the Configuration tab.

- 3. In the navigation pane on the left of the console, click *Dynamic Language Switch*.

On the Dynamic Language Switch window, the languages shipped with WebFOCUS are displayed. By default, English is selected, as shown in the following image.

To turn on the Dynamic Language Switch option on Sign in pages, click on the Enable Dynamic Language check box. The list of available languages to select from will be visible. Choose the languages you wish to enable in the Dynamic Language Switch box.

Client Code Page: 1252

☐ Enable Dynamic Language

<input type="checkbox"/>	Locale	Language Code	Locale Identifier string
<input checked="" type="checkbox"/>	English	en	en_US
<input type="checkbox"/>	English - Australian	au	en_AU
<input type="checkbox"/>	English - Canadian	ca	en_CA
<input type="checkbox"/>	English - United Kingdom	uk	en_GB
<input type="checkbox"/>	French - Canadian	fc	fr_CA
<input type="checkbox"/>	French - Standard	fr	fr_FR
<input type="checkbox"/>	German	de	de_DE
<input type="checkbox"/>	Italian	it	it_IT
<input type="checkbox"/>	Portuguese - Brazilian	br	pt_BR
<input type="checkbox"/>	Spanish	es	es_ES

Save

Cancel

- 4. Select the *Enable Dynamic Language* check box.

Selecting the Enable Dynamic Language check box turns on the Select Language drop down on all WebFOCUS Sign In pages.

To turn on the Dynamic Language Switch option on Sign in pages, click on the Enable Dynamic Language check box. The list of available languages to select from will be visible. Choose the languages you wish to enable in the Dynamic Language Switch box.

Client Code Page: 1252

☒ Enable Dynamic Language

<input checked="" type="checkbox"/>	Locale	Language Code	Locale Identifier string
<input checked="" type="checkbox"/>	English	en	en_US
<input checked="" type="checkbox"/>	English - Australian	au	en_AU
<input checked="" type="checkbox"/>	English - Canadian	ca	en_CA
<input checked="" type="checkbox"/>	English - United Kingdom	uk	en_GB
<input checked="" type="checkbox"/>	French - Canadian	fc	fr_CA
<input checked="" type="checkbox"/>	French - Standard	fr	fr_FR
<input checked="" type="checkbox"/>	German	de	de_DE
<input checked="" type="checkbox"/>	Italian	it	it_IT
<input checked="" type="checkbox"/>	Portuguese - Brazilian	br	pt_BR
<input checked="" type="checkbox"/>	Spanish	es	es_ES

Save

Cancel

The default language (for example, English) is automatically enabled. The default language is the one that was selected during installation.

Once you enable Dynamic Language Switch, you can select any language or languages that you want visible.

The languages available in Dynamic Language Switch are determined by the client code page. To make all localized languages available, use a Unicode code page, such as 65001.

In the following image, Spanish is selected. Because the code page is set to 1252, only languages that use code page 1252 are available.

To turn on the Dynamic Language Switch option on Sign in pages, click on the Enable Dynamic Language check box. The list of available languages to select from will be visible. Choose the languages you wish to enable in the Dynamic Language Switch box.

Client Code Page: 1252

☒ Enable Dynamic Language

<input type="checkbox"/>	Locale	Language Code	Locale Identifier string
<input checked="" type="checkbox"/>	English	en	en_US
<input type="checkbox"/>	English - Australian	au	en_AU
<input type="checkbox"/>	English - Canadian	ca	en_CA
<input type="checkbox"/>	English - United Kingdom	uk	en_GB
<input type="checkbox"/>	French - Canadian	fc	fr_CA
<input type="checkbox"/>	French - Standard	fr	fr_FR
<input type="checkbox"/>	German	de	de_DE
<input type="checkbox"/>	Italian	it	it_IT
<input type="checkbox"/>	Portuguese - Brazilian	br	pt_BR
<input checked="" type="checkbox"/>	Spanish	es	es_ES

Save

Cancel

5. Click Save to save your changes.

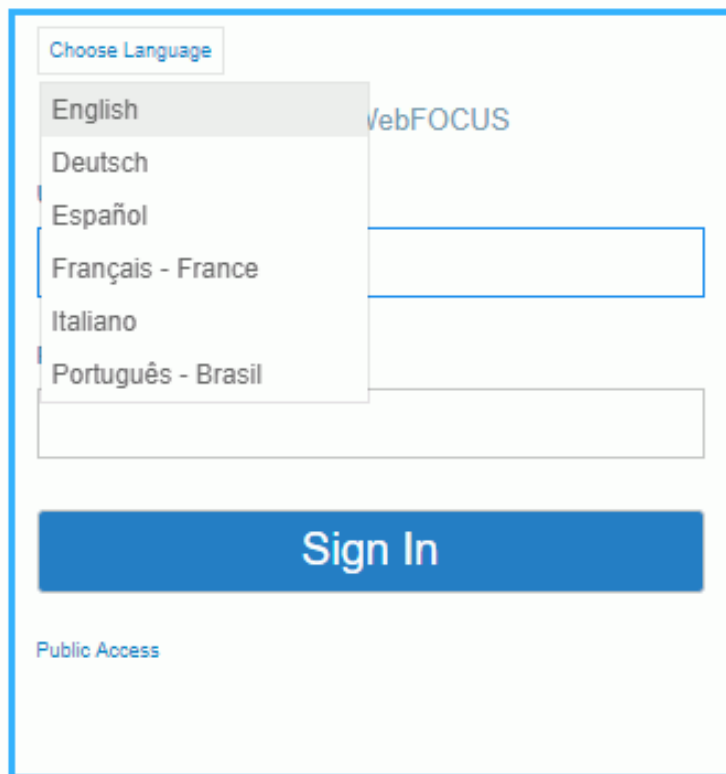
For example, selecting *Spanish* adds the option for Spanish to the Select Language drop-down list on the WebFOCUS Sign In page, as shown in the following image.



Switching Between Languages

After the WebFOCUS administrator enables and customizes the dynamic language switch, you can select from the languages that appear in the Select Language drop-down list on a WebFOCUS Sign In page.

The following image shows a sample drop-down list on the WebFOCUS Sign In page.



The image shows a sample WebFOCUS Sign In page. At the top left, there is a button labeled "Choose Language". Below this button is a drop-down menu with the following options: English, Deutsch, Español, Français - France, Italiano, and Português - Brasil. The "English" option is currently selected and highlighted. To the right of the language menu, the text "WebFOCUS" is visible. Below the language menu, there are two empty input fields. At the bottom of the page, there is a large blue button labeled "Sign In". Below the "Sign In" button, the text "Public Access" is displayed.

Once you select a language, the user interface automatically switches to that language, for example, French, as shown in the following image.



WebFOCUS creates two cookies that hold the language value selected when signing in.

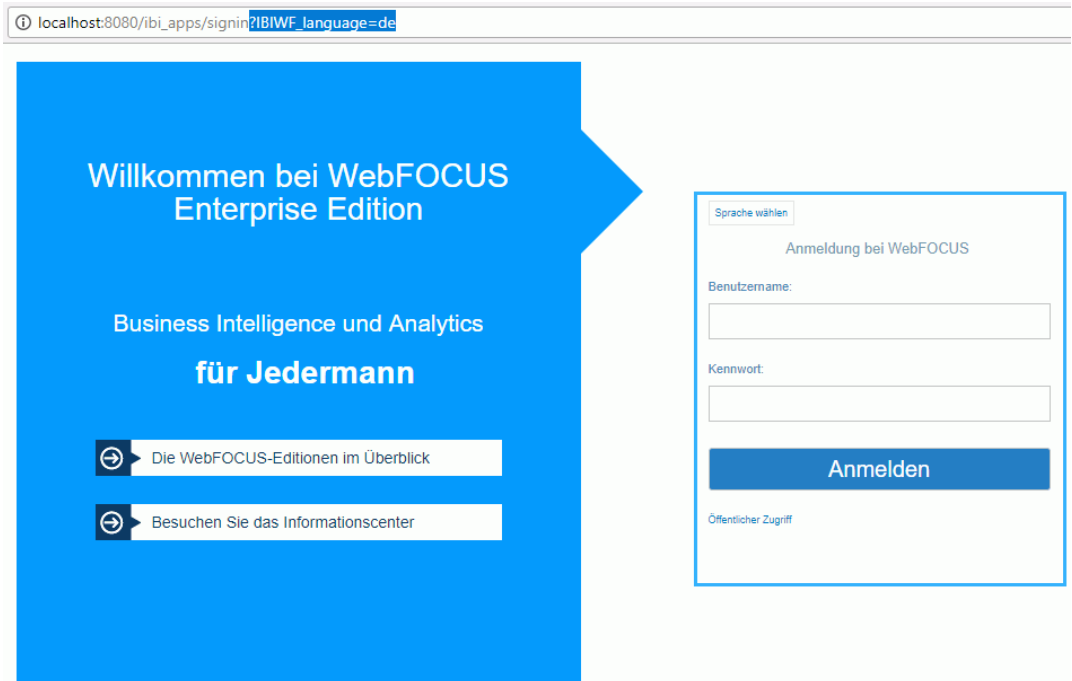
- ❑ The first one is a temporary cookie, `IBIWF_language`, that passes the language value to all WebFOCUS components during the browser session.

Note that if multiple browser sessions are open, with a different language in each one, multiple temporary cookies with different languages may exist on the same user machine.

- ❑ The second is a permanent cookie, `IBIWF_langperm`, that sets the initial language that appears every time a browser session is opened. The language may be different from the operating system language.

Passing the Language Value to the URL

To force a WebFOCUS Sign In page to appear in a specific language prior to user selection, you can add the IBIWF_language parameter to the URL.



Syntax: How to Pass the Language Value to the URL

http://host_name:port_number/application_address?IBIWF_language=xx

where:

host_name

Is the name of the machine on which the WebFOCUS Client is installed.

port_number

Is the port number on which the server is listening.

application_address

Is the startup location of the application.

IBIWF_language

Is the parameter that specifies the language value. See [Values for Supported Languages](#) on page 43.

xx

Is the two-letter value for a valid installed language. en is the default value, for English.

Reference: Values for Supported Languages

The following table lists the values you can specify on the IBIWF_language parameter.

Language	Value
English	en (default)
Chinese (Simplified GB)	zh
Chinese (Traditional Big-5)	tw
French (Canadian)	fc
French (Standard)	fr
German (Standard)	de
Italian	it
Japanese	ja
Portuguese (Brazilian)	br
Spanish	es

Procedure: How to Change the Default Value of IBIWF_language

This procedure controls the initial language in which a sign-in page appears if the browser language is not one of the languages in the ArrayofTopics list in ibimultilanguage.js. If a user selects a different language when signing in, the value of that language is stored in the cookies and takes precedence over the value of the browser language or the default setting.

1. Sign in to the WebFOCUS Administration Console.

Access the WebFOCUS Administration Console in one of two ways:

- ❑ From the Start menu, navigate the app list to *Information Builders* and click *WebFOCUS Administration Console*.

or

- ❑ Enter the following URL in your browser:

`http://host_name:port/ibi_apps`

where:

`host_name`

Is the name of the machine on which the WebFOCUS Client is installed.

`port`

Is the port number on which the server is listening.

The WebFOCUS Sign In page opens. Sign in as an administrator.

The WebFOCUS Home Page opens. Open the *Administrator* menu, point to *Administration*, and click *Administration Console*. The Administration Console opens to the Configuration tab.

2. From the navigation pane, click *Client Settings* in the Application Settings section.
3. Locate the IBI Language parameter, and select a language value.

The default value is en_US.

4. Click *Save* and exit the console.

Creating a Localized Version

You can create additional localized versions of WebFOCUS in addition to the localized versions supported by Information Builders. When you create a localized version, the user interface appears and functions the same as the original version. The only difference is the language displayed in the interface.

In order to configure the WebFOCUS interface to display another language, you must first create resource files by executing the Language Resource Files Utility to automatically install the necessary resource files. In the new resource files you must then translate the English character strings to the new language.

The utility is located in the `drive:ibi\WebFOCUSnn\utilities\lang` directory as `CreateLangRes.bat` in Windows installations, where `nn` is your version of WebFOCUS, and as `CreateLangRes` in UNIX installations.

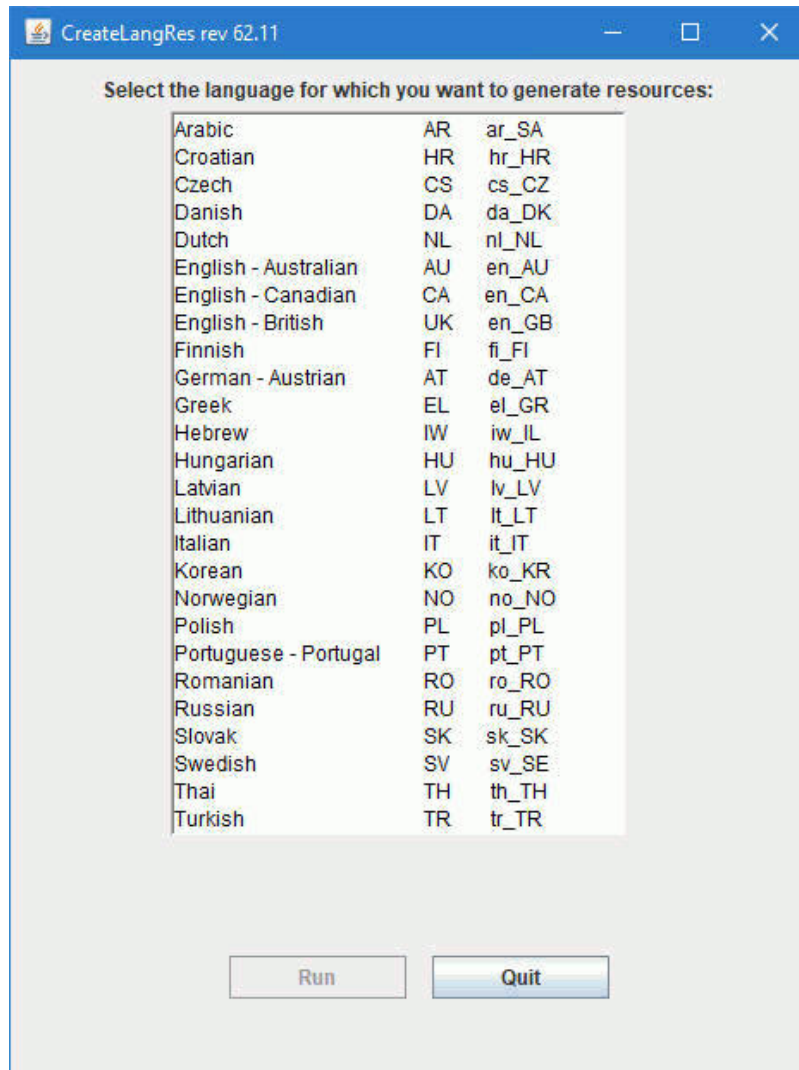
You can install the following language files with English content by running this utility:

Arabic	Italian
Croatian	Korean
Czech	Latvian
Danish	Lithuanian
Dutch	Norwegian
English (Australian)	Polish
English (Canadian)	Portuguese (Portugal)
English (British)	Romanian
Finnish	Russian
German (Austrian)	Slovak
Greek	Swedish
Hebrew	Thai
Hungarian	Turkish

Note: The Austrian German resources are created from the German resources.

The utility provides prompts and the necessary information to guide you during the installation.

Note: The following image is from a Windows version. Similar text and prompts are available from the UNIX version.



Example: Creating a Language Resource File

The following is an example of how to install the German (Austria) language resource files. When you install new files, any existing language resource files are overwritten.

1. Navigate to *drive:\ibi\WebFOCUSnn\utilities\lang*, where *nn* is your version of WebFOCUS, and double-click *CreateLangRes.bat* to execute the utility.
2. Ensure that your web server and application server are stopped, as instructed. Enter *C* to continue.

The CreateLangRes dialog box opens.

3. Select *German - Austrian* from the list of available languages.
4. Ensure that the correct WebFOCUS installation directory is listed, and click *Run*.
5. The utility displays which set of files are being created and the user is notified upon successful completion.

A ConvertLangRes log file is created at *drive:\ibi\WebFOCUSnn\logs*, where *nn* is the version of WebFOCUS that you are using.

Translating a Properties File

For files that must be modified and have the extension *.properties*, the text within the file is structured in pairs. On each line exists a pair, separated by an equal sign (=). On the left side is the original text as processed by WebFOCUS. On the right are the words written as they will be seen by the user. For example:

```
welcome=Welcome
to_the=to the
```

If you want to open an existing translated file to see how it is translated for other languages, you can do so. These files have either a language prefix or suffix added to the file name. For example, a file translated into Spanish has the prefix or suffix *ES*.

Note: All **.properties* files must be encoded as UNICODE. If you want to make a change, you have to use the *native2ascii* java utility. For more information on how to use this utility, go to: <http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/native2ascii.html>

The use of visible NLS characters as part of the Latin-1 encoding is acceptable because the Latin-1 area of ASCII and UNICODE are transparent (except for the euro symbol). If the file includes a euro symbol, or any non-Latin-1 character (such as Hebrew, Chinese, or Japanese), it must be transcoded using the *native2ascii* tools that Java JDK provides. For more information about the usage of *native2ascii*, please refer to the Oracle JDK documentation.

Procedure: How to Localize a Properties File

This example illustrates how to translate the text on the right side of the equal sign to read as it should in the localized language:

WebFOCUS text=localized text

where:

WebFOCUS text

Is the text processed by WebFOCUS. This text should not be modified.

localized text

Is the original English text translated into the localized language.

Example: Translating Lines in a Properties File

The following is an example of the beginning of a properties file that has been translated into Spanish:

```
# $Revision: 1.55 $:
# US_Revision: 1.110
# @(#)CasterSchedule.properties
#
# Resource strings for ReportCaster

##### Main Window #####
### Main Frame ###
Main.Title=ReportCaster
### Main Label ###
Main.Label=Desarrollo y administración de ReportCaster
Main.Icon=reportcaster.gif
Main.Tree.Label=Carpetas

Main.User.page=RCSchedulePage.htm
##### MAIN TREE #####
Main.Tree.Node.root=Elementos de programación
Main.Tree.Node.Root.icon=schedule.gif
Main.Tree.Node.schedules=Elementos de programación
Main.Tree.Node.Schedules.icon=single_user.gif
Main.Tree.Node.Root.page=RCScheduleMain.htm

Main.Page.Schedules=Planificaciones
Main.Page.Library=Librería de informes
Main.Page.Admin=Gerencia y configuración de usuario
```



```
##### MAIN TABLE #####
Main.Table.Header.ACTIVE.tip=Clasificar por: Estado activo
Main.Table.Header.NOTIFY_FLAG.tip=Clasificar por: Indicador de notificación
Main.Table.Header.METHOD_CODE.tip=Clasificar por: Método de distribución
Main.Table.Header.SCHEDULEID.tip=Sort by: Schedule ID
Main.Table.Header.JOBDESC.tip=Clasificar por: Descripción de trabajo
Main.Table.Header.NEXTRUNTIME.tip=Clasificar por: Próximo tiempo de
ejecución
```

Translating a JavaScript Resource File

In JavaScript resource files used by WebFOCUS, each .put statement contains two statements enclosed in double quotation marks ("). The following is an example of a JavaScript .put statement:

```
worptrans.put("Default Report","Default Report");
```

Procedure: How to Localize a JavaScript Resource File

In each .put command, translate the second statement in double quotation marks (") into the localized language, as follows:

```
command.put("WebFOCUS text","localized text");
```

where:

command

Is the JavaScript .put command. This is not modified.

WebFOCUS text

Is the first statement in the parentheses, enclosed in quotation marks. This statement is for WebFOCUS processing, and should not be modified.

localized text

Is the WebFOCUS text translated into the localized language.

Example: Translating Lines in a JavaScript Resource File

The following is an example of a JavaScript resource file translated for a localized French version:

```
/*
** $Revision: 1.41 $
** US Revision: 1.59
*/
var FRranls_js_Revision = "$Revision: 1.41 $";

// *****
// ** WebFOCUS Report Assist NLS Javascript for IE
// *****

function createRATransHashTable()
{
    var rpTrans = new ibihash(mapIbiUrl);
    rpTrans.put("rpassstieTitleString","Générateur de Rapport de WebFOCUS");
    rpTrans.put("haaboutRevision","Version 5 Révision 2.0");
    rpTrans.put("haaboutOKButton","/ibi_html/javaassist/intl/FR/FRHafcokon.gif");
    rpTrans.put("haaboutTableGIF","/ibi_html/javaassist/ibi/html/assist/HAabout.gif");
    rpTrans.put("haaboutGraphGIF","/ibi_html/javaassist/ibi/html/assist/hagabout.gif");
    rpTrans.put("ColumnTitleString","Surlignez les champs et cliquez AJOUTER en mode
        Liste ou glissez-déposez les champs en mode Structure");
    rpTrans.put("ColumnOptionsTitleString","Surlignez le champ que vous voulez
        personnaliser");
}
```

Translating a Text File

Text files used by WebFOCUS contain a revision number at the top of the file, and below this, pairs of text. Each pair of text has two parts: the first is information passed to WebFOCUS followed by a comma, and the second is the text viewed by a user enclosed in double quotation marks ("). The following is an example of lines from a text file as used in WebFOCUS:

```
Sort, "Sort"
Footing, "Footing"
```

Procedure: How to Localize a Text File

Translate any text in double quotation marks (") into the localized language, as follows:

```
WebFOCUS text, "translated text"
```

where:

```
WebFOCUS text
```

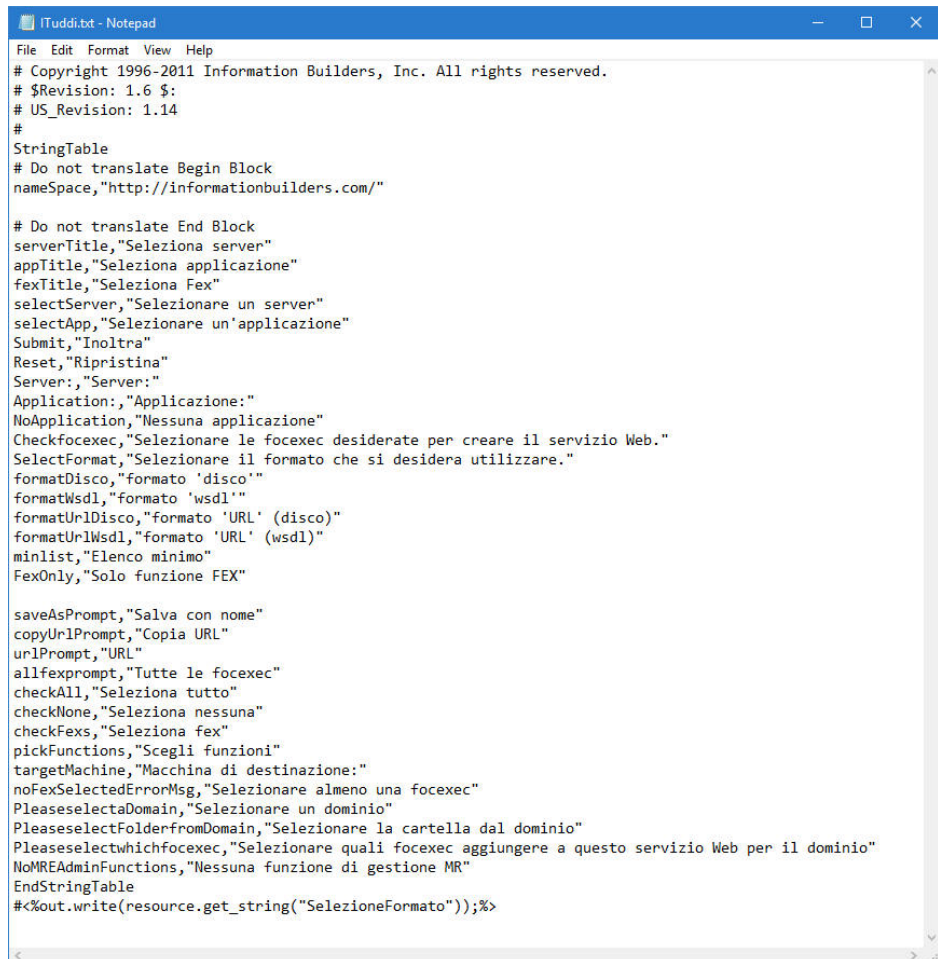
Is the first statement on the line, before the comma (.). This statement is for WebFOCUS processing, and should not be modified.

translated text

Is the second statement in the line, translated into the local language and enclosed in double quotation marks (").

Note: You can only translate text within double quotation marks that fall within the #Do translate Begin Block and the #Do translate End Block lines. Anything within #Do Not Translate Begin Block and #Do Not Translate End Block will cause WebFOCUS to malfunction.

For example, in the following image you can translate everything in double quotation marks (") following the #Do translate Begin Block.



```

ITuddi.txt - Notepad
File Edit Format View Help
# Copyright 1996-2011 Information Builders, Inc. All rights reserved.
# $Revision: 1.6 $:
# US_Revision: 1.14
#
StringTable
# Do not translate Begin Block
nameSpace,"http://informationbuilders.com/"

# Do not translate End Block
serverTitle,"Seleziona server"
appTitle,"Seleziona applicazione"
fexTitle,"Seleziona Fex"
selectServer,"Selezionare un server"
selectApp,"Selezionare un'applicazione"
Submit,"Inoltra"
Reset,"Ripristina"
Server:,"Server:"
Application:,"Applicazione:"
NoApplication,"Nessuna applicazione"
Checkfocexec,"Selezionare le focexec desiderate per creare il servizio Web."
SelectFormat,"Selezionare il formato che si desidera utilizzare."
formatDisco,"formato 'disco'"
formatWsd1,"formato 'wsdl'"
formatUrldisco,"formato 'URL' (disco)"
formatUr1Wsd1,"formato 'URL' (wsdl)"
minlist,"Elenco minimo"
FexOnly,"Solo funzione FEX"

saveAsPrompt,"Salva con nome"
copyUr1Prompt,"Copia URL"
ur1Prompt,"URL"
allfexprompt,"Tutte le focexec"
checkAll,"Seleziona tutto"
checkNone,"Seleziona nessuna"
checkFexs,"Seleziona fex"
pickFunctions,"Scegli funzioni"
targetMachine,"Macchina di destinazione:"
noFexSelectedErrorMsg,"Selezionare almeno una focexec"
PleaseselectaDomain,"Selezionare un dominio"
PleaseselectFolderfromDomain,"Selezionare la cartella dal dominio"
Pleaseselectwhichfocexec,"Selezionare quali focexec aggiungere a questo servizio Web per il dominio"
NoMREAdminFunctions,"Nessuna funzione di gestione MR"
EndStringTable
#<%out.write(resource.get_string("SelezioneFormato"));%>

```

Example: Translating a Text File

The following is an example of a text file modified for a localized French version:

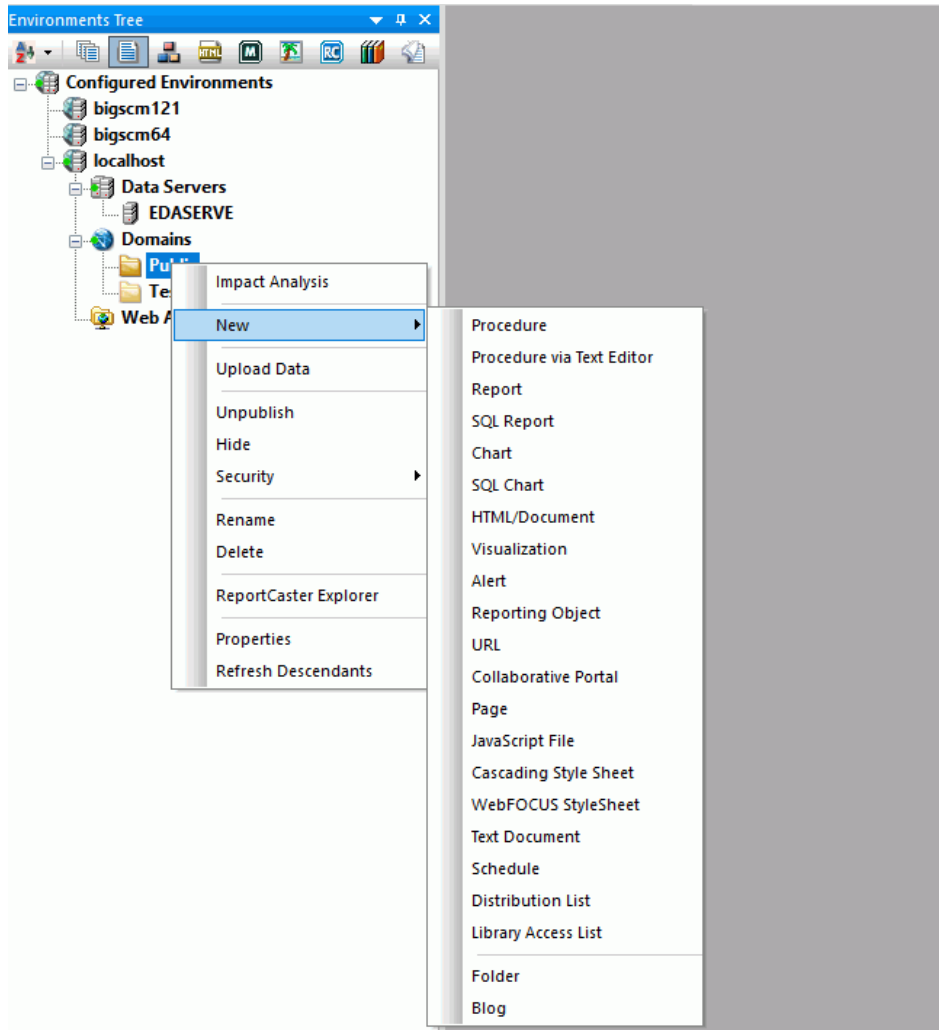
```
# Copyright 1996-2011 Information Builders, Inc. All rights reserved.
$Revision: 1.21 $:
# US_Revision: 1.14
StringTable
# Do not translate Begin Block
nameSpace,"http://informationbuilders.com/"

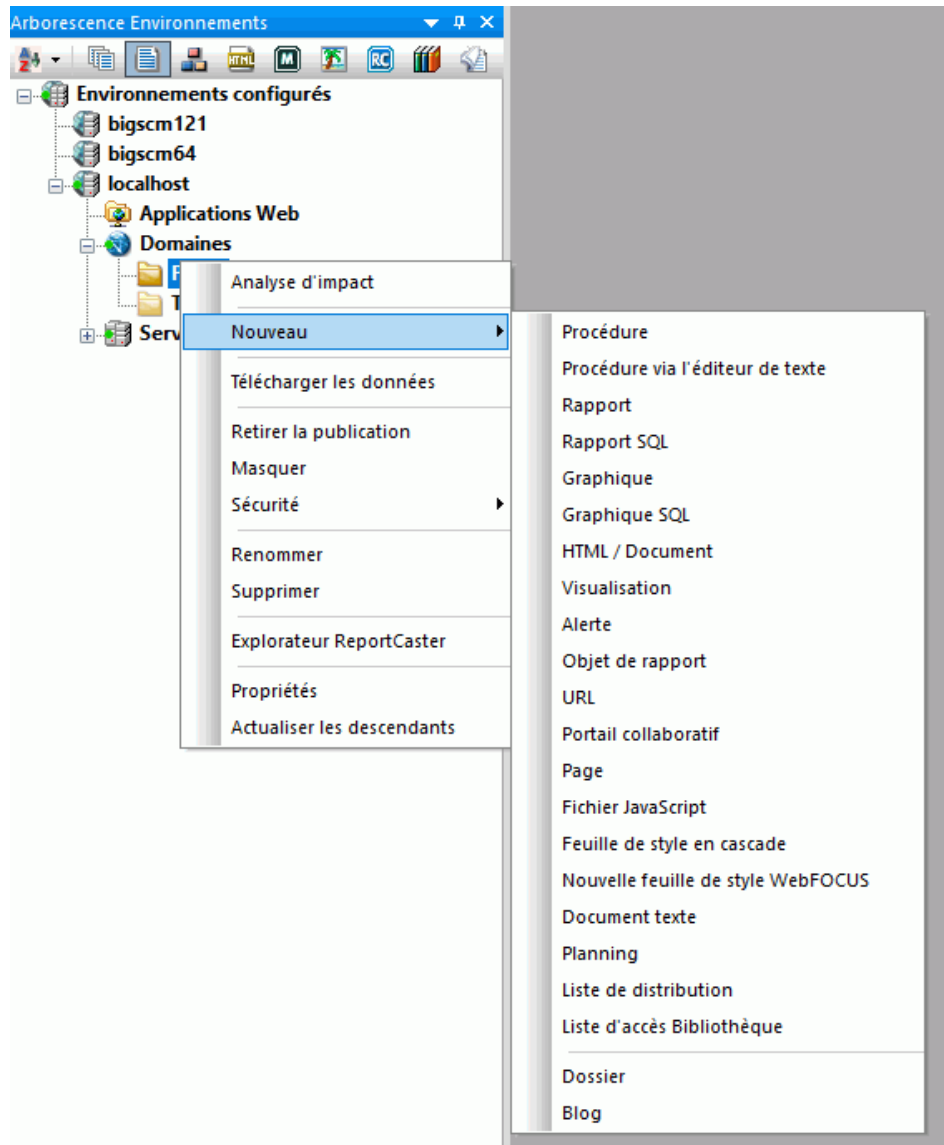
# Do not translate End Block
serverTitle,"S\u00e9lectionnez le serveur"
appTitle,"S\u00e9lectionnez l'application"
fexTitle,"S\u00e9lectionnez les Fex"
selectServer,"Veuillez s\u00e9lectionner un serveur"
selectApp,"Veuillez s\u00e9lectionner une application"
Submit,"Soumettre"
Reset,"R\u00e9initialiser"
Server:,"Serveur:"
Application:,"Application:"
NoApplication,"Aucune application"
Checkfocexec,"S\u00e9lectionnez les focexec(s) souhait\u00e9s pour cr
\u00e9er le service web."
SelectFormat,"S\u00e9lectionnez le format que vous voulez utiliser."
formatDisco,"format 'disco'"
formatWsdL,"format 'wsdl'"
formatUrlDisco,"format 'URL' (disco)"
formatUrlWsdL,"format 'URL' (wsdl)"
minlist,"Liste minimum"
FexOnly,"Fonction FEX uniquement"
```

Using a Localized Version

The program interface for a localized version appears with text in the local language. The functionality of the program interface is the same for any localized version. The buttons and tabbed dialog box options are always in the same position—only the text changes for each local language.

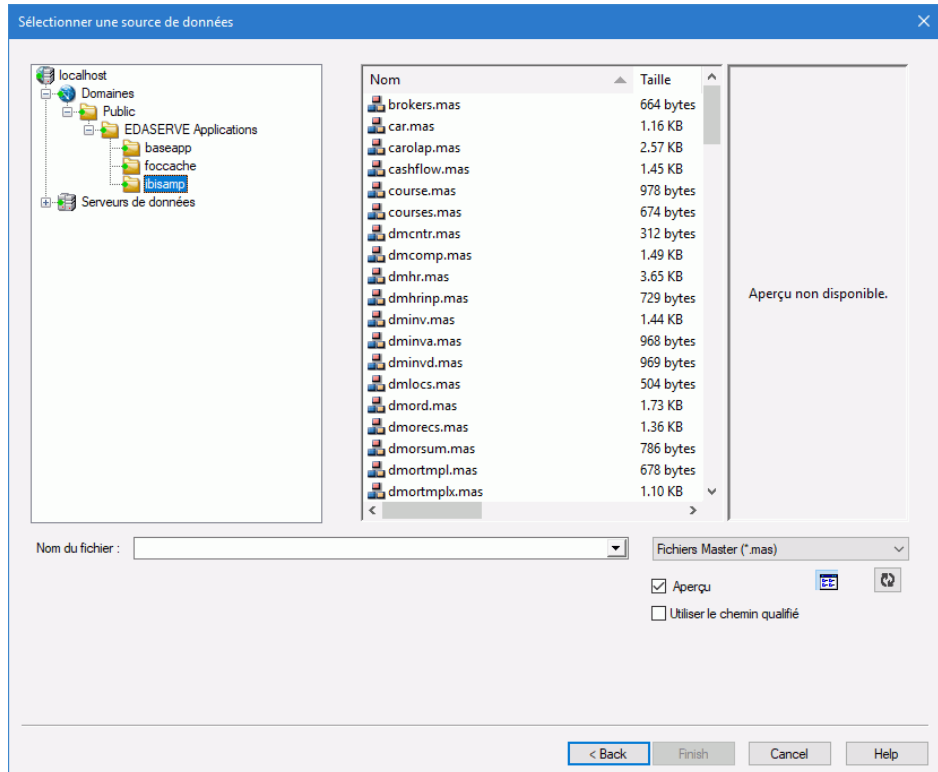
For example, the following is a comparison of the WebFOCUS App Studio shortcut menu in English (top) and French (bottom). The options for choosing assistants and other actions are in the same position, but the text appears in the local language.





The bottom image is from the French localized version. The names of the App Studio facilities are translated so that you can work efficiently with the various localized versions. The assistants and other utilities also appear in the local language.

In the following window, the names for WebFOCUS data sources, such as brokers and car, are not translated. However, they may contain data with national characters, such as French accented letters or German umlauts.



The following InfoAssist+ Report mode window is from the French localized version and uses the French localized wf_retail_lite sample file. The interface text appears in the local language, and the column headers do as well because the synonym has localized column titles. However, a procedure has the same syntax when created in any localized version.

Données - wf_retail_lite

Requête

Aperçu instantané (500 Enregistrements)

Champs de recherche

Requête de variables

Mesures

Ventes

Expéditions

Dimensions

En relation_Ventes

Produit

Produit,Catégorie

Produit,Sous-catégorie

Modèle

Valeurs

Connexe_expéditions

Magasin

Client

Rapport (wf_retail_lite)

Somme

Quantité,Vendue

Coût des marchandises

Vertical

Produit,Catégorie

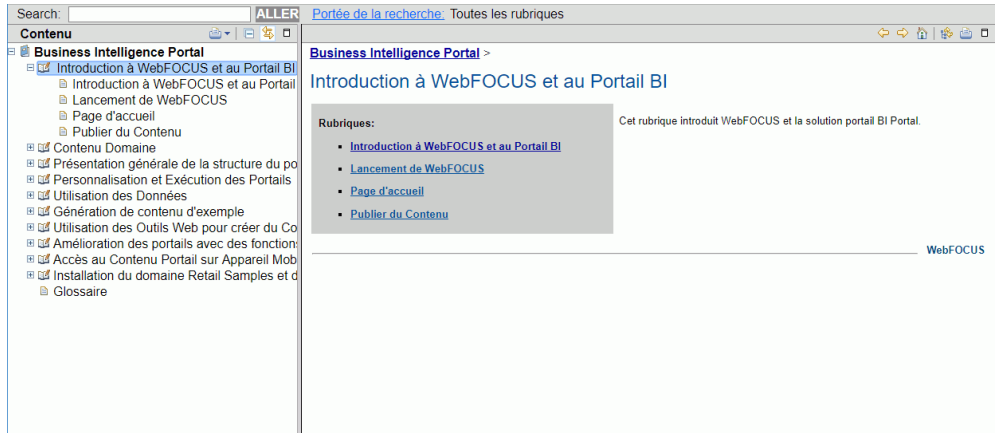
Horizontal

Filtre

Produit	Quantité	
Catégorie	Vendue	Coût des marchandises
Accessories	2,034	\$342,877.00
Camcorder	1,956	\$453,205.00
Computers	865	\$109,281.00
Media Player	3,140	\$779,593.00
Stereo Systems	4,679	\$857,042.00
Televisions	362	\$227,820.00
Video Production	887	\$180,540.00

Accessing Local Language Online Help

Online Help in a localized version functions identically to the English version of Online Help and covers the same topics. However, all the text appears in the local language. The following WebFOCUS Online Help window is from the French localized version.



Working With an Unfamiliar Local Language

Most operating systems are now available in localized versions for many languages. In some cases, such as UNIX, you specify the locale for a single version of the operating system. Localized versions of operating systems work the same way as Information Builders localized versions, meaning they are functionally equivalent and positional. If you are familiar with one localized version of the product, you can use any other localized version, even if you are not familiar with the language.

The major difference between the various localized versions is the provision of special Input Method Editors (IMEs) for Far Eastern Chinese, Japanese languages and other scripts. The Far Eastern CJK languages use a more complicated method of generating ideographic input than simply pressing a single key for each written symbol. The CJK languages generally use IMEs, which are typically included in the computer operating system. There are several standard input methods for each of the CJK languages, and operating systems often support more than one. Refer to your operating system documentation for details.

Interpreting Local Language Error Messages

When you use a localized version, most error messages appear in the local language. You must also configure the WebFOCUS Reporting Server NLS settings to enable localized error messages generated by the server. For more information on enabling server error messages, see [Localizing the WebFOCUS Reporting Server Console](#) on page 71.

The following numbered server error messages (in the form FOC nnn) are from the English version of WebFOCUS.

```
{FOC014} THE NUMBER OF ACROSS SORT FIELDS EXCEEDS THE MAXIMUM  
{FOC009} INCOMPLETE REQUEST STATEMENT  
BYPASSING TO END OF COMMAND
```

The following are the same error messages in the Spanish localized version, generated under identical circumstances.

```
{FOC014} EL NUMERO DE CAMPOS DE CLASIFICACION ACROSS EXCEDE EL MAXIMO  
{FOC009} SENTENCIA DE PETICION INCOMPLETA:  
{INF32074} BYPASSING TO END OF COMMAND
```

Since numbered error messages always refer to the same condition, if you are less familiar with one language, you can check the error message number in the documentation of a language that you are more familiar with to find a detailed description of the problem.



Chapter 3

Configuring the WebFOCUS Reporting Server for NLS

This topic describes the steps for configuring a WebFOCUS Reporting Server for National Language Support (NLS) on Windows, UNIX, IBM i, and z/OS using the supplied web-based Consoles.

Configuration through the web-based consoles provides standard character transcoding, monocasing, and sorting and is sufficient for most enterprises.

It also describes how to localize the WebFOCUS Reporting Server Web Console.

Note: This manual uses the term "IBM i" generically to refer to all IBM i, i5/OS, and OS/400 releases.

In this chapter:

- ❑ [Configuring a WebFOCUS Reporting Server for NLS](#)
 - ❑ [Localizing the WebFOCUS Reporting Server Console](#)
 - ❑ [Configuring National Language Support for the Data Management Console](#)
-

Configuring a WebFOCUS Reporting Server for NLS

You can use the NLS (National Language Support) Configuration Wizard to select the code page for the data sources your server will access. Access the NLS Configuration Wizard from the Workspace tab in the WebFOCUS Reporting Server Console.

A code page is the computer representation of a character set. Each written symbol in a language is assigned a unique number, usually expressed in hexadecimal notation. A code page has a unique identification number to distinguish the operating system and language or languages to which it applies.

The WebFOCUS Reporting Server maintains a default code page generation file. After you configure the server using the WebFOCUS Reporting Server Console, WebFOCUS updates the default file with the code page selected for the server and with the associated client code pages. The updated file, which contains the new code page values and the default values, is used to generate the transcoding, monocasing, and sorting tables. For the default code page values, see [Default Code Page Generation File](#) on page 68.

To configure the server for NLS, you must:

1. Determine which code page you need.

2. Configure the WebFOCUS Reporting Server for NLS.

Reference: Configuring National Language Support on IBM i

If you are running OS/400 V5R1 or a later version of IBM i, the WebFOCUS installation procedure prompts you for an initial code page. Your response ensures that the necessary files are unloaded properly, but does not result in actual configuration of the WebFOCUS Reporting Server and WebFOCUS Client.

You must configure the WebFOCUS Reporting Server and WebFOCUS Client for the specific code page and other desired NLS features using the WebFOCUS Reporting Server Console and WebFOCUS Administration Console, as described in this content.

For information on the WebFOCUS installation procedure for OS/400 and IBM i, contact your Information Builders representative.

Procedure: How to Configure the WebFOCUS Reporting Server for NLS

1. Access the WebFOCUS Reporting Server Web Console, and sign in with administrative privileges.

To access the Web Console, type the following URL into your web browser address bar:

`http://host:port`

where:

`host`

Is the host name or IP address of the machine running the WebFOCUS Reporting Server.

`port`

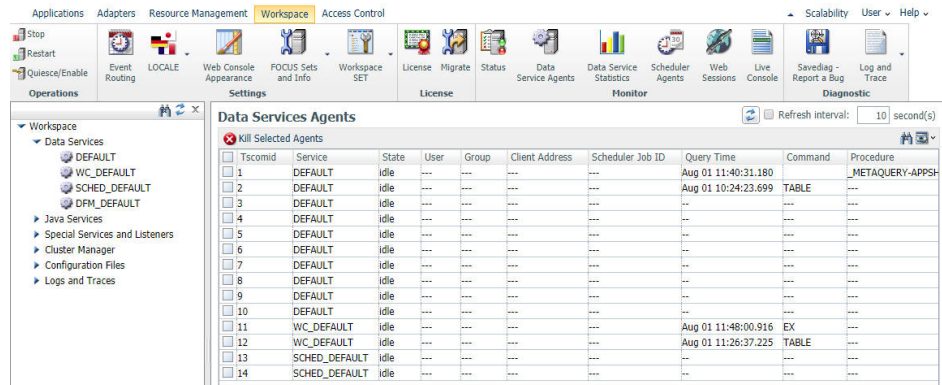
Is one number higher than the port number specified when installing the server. For example, if at installation you accepted the default port number of 8120, to access the console you would specify port number 8121.

If you installed the server on a machine with the host name MyServer and chose the base port 9190, to access the console you would enter `http://MyServer:9191`.

Alternatively, with WebFOCUS running on Windows, you can also access the Web Console in the following ways:

- ☐ From the Start menu, expand the *Information Builders* app and click *Web Console*.
 - ☐ In App Studio, select *Reporting Server Console* from the WebFOCUS Administration menu. The console for the project-based development environment opens.
2. In the Web Console's menu bar, click *Workspace*.

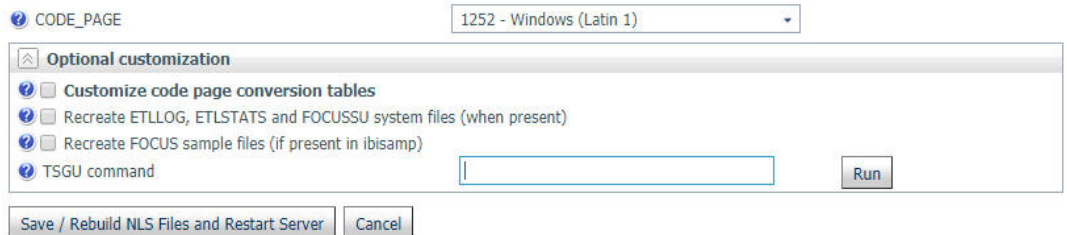
The Workspace tab opens, as shown in the following image.



- On the Workspace tab, in the Settings group, click *LOCALE*, and then click *Configuration Wizard*.

The NLS Configuration Wizard pane opens, as shown in the following image.

NLS Configuration Wizard



The following steps provide information on modifying the settings in the NLS Configuration Wizard.

- Select a code page from the CODE_PAGE drop-down menu. For more information about code pages, see [Understanding Code Pages](#) on page 17.

The default code page for the WebFOCUS Reporting Server is 1252.

- Select the *Customize code page conversion tables* check box to display a list of code pages and descriptions for the possible data sources accessed by the server or the client code page.

Several code pages are always selected by default and cannot be deselected. These are: 37–IBM EBCDIC United States; 137–U.S. English/Western Europe (Latin 1); 437–U.S. English; 1047–IBM EBCDIC Open Systems (Latin 1); 1252–Windows (Latin 1); 65001–Unicode (UTF-8); 65002–Unicode (UTF-EBCDIC). Other code pages may also be selected by default, depending on the LANG attribute chosen in the LOCALE settings.

The NLS Configuration Wizard creates conversion tables for these code pages to make them available for subsequent conversion.

Select any additional code pages for which you want tables to be created. For example, if you want to designate a particular code page to be generated in synonyms for flat (fixed) files or VSAM files, you must already have created a code page conversion table using the Customize code page conversion table option during NLS configuration. For related information, see the *Synonym Creation Parameters* section in the adapter chapter specific to your data source in the *Adapter Administration* manual.

6. Select the *Recreate ETLLOG, ETLSTATS and FOCUSU system files (when present)* check box if you wish to load internal files in the server's code page (as defined in the CODE_PAGE field).
This is recommended if the compatibility of the new and previous code pages is unknown.
7. Select the *Recreate FOCUS sample files (if present in ibisamp)* check box if you wish to load the sample files (which can be generated by creating a new tutorial in an application folder) in the server's code page (as defined in the CODE_PAGE field).
8. You can run a TSGU command from the NLS Configuration Wizard if you wish. Type the command into the TSGU command text box and click *Run*. A new tab will open with the output.

TSGU (Translation Services Generation Utility) is a program that you can use to configure NLS settings. Usually, this program is not required since the NLS Configuration Wizard takes care of the configuration transparently. However, under the following circumstances, you may need to enter and execute the TSGU command manually:

- ☐ To create PDFXTBL. When TYPE 1 fonts should be included in PDF format file, the TYPE 1 font transcoding table must be generated. The syntax is:

PDFX {font-name}

- ☐ To create PDFYTBL. When Unicode fonts should be used in PDF format file, the special font information tables (PDFYTBL) must be generated. The syntax is:

KTBL {pdfy-name}

For example, KTBL PDFX generates all the special font information files.

If required, enter the command in the TSGU input box and click the blue arrow to execute it. The results are displayed in a separate window.

Procedure: How to Change LOCALE Settings on the WebFOCUS Reporting Server

You can use the Change Core Engine Settings in `edasprof.prf` page to set LOCALE settings, such as the language and number, currency, and date formatting on the WebFOCUS Reporting Server. You can also manually enter these values in the `edasprof.prf` file, which you can access under *Configuration Files* in the Web Console Workspace tab in the navigation pane.

To use the Change Core Engine Settings in `edasprof.prf` page to set your LOCALE settings, complete the following steps:

1. Open the WebFOCUS Reporting Server Web Console with administrative privileges.

To access the Web Console, in the address bar of your browser, enter the following URL:

`http://host:port`

where:

host

Is the host name or IP address of the machine running the WebFOCUS Reporting Server.

port

Is one number higher than the port number specified when installing the server. For example, if at installation you accepted the default port number of 8120, to access the console you would specify port number 8121.

If you installed the server on a machine with the host name `MyServer` and chose the base port 9190, then to access the console you would enter `http://MyServer:9191`.

Alternatively, with WebFOCUS running on Windows, you can also access the Web Console in the following ways:

- ☐ From the Start menu, expand the *Information Builders* app and click *Web Console*.
- ☐ In App Studio, select *Reporting Server Console* from the WebFOCUS Administration menu. The console for the project-based development environment opens.

2. In the Web Console menu bar, click *Workspace*.

The Workspace tab opens.

3. On the ribbon, in the Settings group, click *LOCALE* and then click *LOCALE (Language, Numbers, Currency, Dates)*.

The Change Core Engine Settings in the `edasprof.prf` page opens.

The following steps describe how to modify the LOCALE settings.

4. Select your language from the LANGUAGE drop-down list.

5. You can select a collation method from the COLLATION drop-down menu. Collation determines the ordering and matching of all language elements that involve comparison of two alphanumeric values, such as for sorting data alphabetically.

You can select one of the following options:

- ☐ **CODEPAGE.** CODEPAGE bases the collation sequence on the code page in effect, and is case-sensitive. This is the default.
 - ☐ **BINARY.** Bases collation on binary values. This is equivalent to CODEPAGE in most cases, except for Danish, Finnish, German, Norwegian, and Swedish localized environments using an EBCDIC code page.
 - ☐ **SRV_CS.** Bases collation on the LANGUAGE setting and is case-sensitive.
 - ☐ **SRV_CI.** Bases collation on the LANGUAGE setting and is case-insensitive.
6. You can set a number format from the CDN drop-down menu. These settings affect numbers one thousand and higher and numbers containing decimals.
 7. You can set the currency code to display from the CURRENCY_ISO_CODE drop-down menu. This code will display if the currency symbol cannot be displayed by the defined code page.
 8. You can determine where to display the currency symbol or ISO code using the CURRENCY_DISPLAY drop-down menu.
 9. You can use the CURRENCY_PRINT_ISO setting to determine when to use the ISO code in place of the currency symbol.

By default, the ISO code will only be displayed when the currency symbol cannot be displayed. You can also set the ISO code to always be used, including in place of the currency symbol, or to never be used, even when no currency symbol is available.
 10. You can specify the order of date field components using the DATE_SEPARATOR drop-down menu.

By default, this is determined by the format of the individual field, but you can use this option to overwrite it.
 11. You can specify a character to separate date components from the DATE_SEPARATOR drop-down menu.

By default, the separator character is taken from the USAGE specifications of the date field, but you can use this option to overwrite it.
 12. You can specify a character to separate time components from the TIME_SEPARATOR drop-down menu.

Your options are a dot (.) or a colon (:).
 13. You can change the business day specification for use in date and time functions from the default Monday through Friday setting by modifying the value in the BUSDAYS text box.

To change the specified business days, replace any of the days of the week, represented by the first letters of each day in order (SMTWTFS), with an underscore.

For example, changing the BUSDAYS value to `__TWTFS` would specify that Tuesday through Saturday are business days. You can use the drop-down menu to reset the value to the default.

14. You can specify a holiday file using the HDAY text box to determine which days to treat as non-business days for date functions.

The value must be two, three, or four letters or numbers in the name of a .err file used to specify holidays. The name of the file must be `HDAYxxxx.err`. `xxxx` is the value that is typed into the HDAY field.

The file must be a text file containing the date of a holiday in YYMD format with no separators, followed by a space and the name of the holiday, one per line. Dates must be in chronological order. For example:

```
20161231 New Year's Eve
20170101 New Year's Day
.
.
.
20171231 New Year's Eve
20180101 New Year's Day
```

For more information, see *Specifying Holidays* in the *Using Functions* manual.

15. You can use the WEEKFIRST drop-down menu to change the first day of the week as it applies to the WEEK and WEEKDAY components of date and date and time functions.
16. Click **Save** to save your settings, and then click **Restart** from the Operations group on the ribbon to restart your server and apply any changes.

A message indicates that the server is restarting. At this point, the server is creating the necessary transcoding, sorting, and monocasing tables for the selected code page.

The Web Console is then redisplayed.

Procedure: How to Edit the NLS Configuration File

The NLS configuration file controls the value of the code page and the language of server error messages.

For Windows, UNIX, IBM i, and z/OS (HFS deployment). The NLS configuration file, `nlscfg.err`, is located in the `etc` directory under the configuration directory. For example, on Windows: `drive:\ibi\sr77\wfs\etc`.

For z/OS (PDS deployment). The NLS configuration file, NLSCFG, is a member of the data set *qualif.EDAMSG.DATA*.

Alternatively, you can access the NLS configuration file from the Web Console.

1. In the Web Console menu bar, click *Workspace*.

The Workspace tab opens.

2. In the navigation pane, expand the *Configuration Files* folder and then the *Miscellaneous* folder, and open *LOCALE - nlscfg.err*.

The Edit LOCALE Configuration File pane opens.

3. Enter the desired parameters. For the settings you can control, see [NLS Configuration File Parameters](#) on page 66.
4. Click the *Save and Restart Server* button.

A message indicates that the server is restarting. Click *OK* to continue. At this point, the server is creating the necessary transcoding, sorting, and monocasing tables for the code page selected.

Example: Editing the NLS Configuration File

The following Windows example configures the server to use code page 1252.



Reference: NLS Configuration File Parameters

This file is called:

- ❑ **nlscfg.err** on Windows, UNIX, IBM i, and z/OS (HFS deployment).
- ❑ **NLSCFG** on z/OS (PDS deployment).

Note that you can manually add a few NLS parameters in the NLS configuration file that are not available from the Web Console NLS Configuration Wizard.

`CODE_PAGE = code_page`

Is the code page of the data sources accessed by the server.

- ❑ For Windows and UNIX, the default value is 1252.

- ❑ For IBM i and z/OS, the default value is 37.

`ADDLANG = language`

Is the language abbreviation code (LANGAB) or language name (LANGNM) from the ? LANG SET table. For details, see [Language and Default Code Pages for Windows, UNIX, and Linux](#) on page 74 or [Language and Default Code Pages for IBM i and z/OS](#) on page 76.

`EXL2KLANG = excel_language`

Is the language of Excel running in the end user browser. The WebFOCUS Reporting Server needs to know the language in order to correctly format output returned to Excel. Possible values are:

- ❑ **ENG** for English. ENG is the default value.
- ❑ **FIN** for Finnish.
- ❑ **FRE** for French.
- ❑ **GER** for German.
- ❑ **JPN** for Japanese.
- ❑ **NOR** for Norwegian.
- ❑ **POL** for Polish.
- ❑ **PRC** for Simplified Chinese.
- ❑ **SPA** for Spanish.
- ❑ **ROC** for Traditional Chinese.

`EXL2K_DEFCOLW = value`

Is the value used by the WebFOCUS Reporting Server for formatting output returned to Excel. It is a fixed value for each language.

WebFOCUS automatically uses the correct value for certain languages, and you do not need to do anything:

- ❑ For English, Finnish, Japanese, Polish, Simplified Chinese, and Traditional Chinese, WebFOCUS uses the correct value of 8 (the default).
- ❑ For French, German, Norwegian, and Spanish, WebFOCUS uses the correct value of 10.

Determining Which Code Page You Need

Today, most Database Management Systems (DBMSs) have ways to refer to the code page associated with the data. See your database administrator to get this information so that you can select the corresponding server code pages to interpret the data correctly.

For Windows, UNIX, Linux, IBM i, and z/OS (HFS deployment):

By default, the server uses code page 1252 (AMENGLISH) on Windows, UNIX, and Linux and code page 37 (AMENGLISH) on IBM i and z/OS (HFS deployment).

For a list of supported code pages, refer to the file `cpxcptbl.nls`, which resides in the `nls` directory under the installation directory (for example, on Windows, `drive:\ibi\srv77\home\nls`).

If the code page needed to interpret your data is not listed in the supported code page file, `cpxcptbl.nls`, contact your WebFOCUS representative.

For z/OS (PDS deployment):

By default, the server uses U.S. EBCDIC code page 37 (AMENGLISH) on z/OS.

For a list of supported code pages, refer to the member `CPXCPTBL` in `qualif.EDACTL.DATA`.

If the code page needed to interpret your data is not listed in the supported code page file, `CPXCPTBL`, contact your WebFOCUS representative.

Reference: Default Code Page Generation File

The following is the default code page generation file:

❑ **cpcodepg.nls**. For Windows, UNIX, Linux, IBM i, and z/OS (HFS deployment).

❑ **CPCODEPG**. For z/OS (PDS deployment).

```
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)
CP65001    A UTF8    Unicode (UTF-8)
```

Reference: Supported Code Page File

The following is the supported code page file:

❑ **cpxcptbl.nls** for Windows, UNIX, IBM i, and z/OS (HFS deployment).

❑ **CPXCPTBL** for z/OS (PDS deployment).

```

* The active Code Page information is represented by lines whose first 2
* columns are 'CP'.
* Valid DBCSID : SOSI,SJIS,EUC,KPC,KKS,TPC,TBIG5,SOSIF,SOSIH
*   C - Character type : E = EBCDIC
*                       A = ASCII
*   DBCS - DBCS ID      : SBCS = Single Byte Character Set
*                       SOSI = IBM MF
*                       SJIS,EUC,SOSIF,SOSIH = Japanese
*                       TPC,TBIG5,TNS,TTEL = Taiwanese
*
*   note: NRC Set = National Replacement Character Set
*         MF      = Mainframe
*

```

Codepage	C DBCS	Description
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
CP00281	E SBCS	IBM MF Japanese English
CP00284	E SBCS	IBM MF Spain/Latin America
CP00285	E SBCS	IBM MF United Kingdom
CP00297	E SBCS	IBM MF France
CP00420	E SBCS	IBM MF Arabic Bilingual
CP00424	E EBCH	IBM MF Israel(Hebrew)
CP00850	A SBCS	IBM PC Multinational
CP00856	A SBCS	IBM PC Hebrew
CP00860	A SBCS	IBM PC Portugal
CP00861	A SBCS	IBM PC Iceland
CP00862	A ASCH	IBM PC Israel
CP00863	A SBCS	IBM PC Canadian French
CP00864	A SBCS	IBM PC Arabic
CP00865	A SBCS	IBM PC Nordic
CP00869	A ASCG	IBM PC Greece
CP00871	E SBCS	IBM MF Iceland
CP00875	E EBCG	IBM MF Greece
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 (Netherlands) 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
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 )
CP00935  E SOSI  PRC IBM MF Extended (cp836+cp837)
CP00946  A CPC   PRC IBM PC Extended (cp1042+cp928)
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 (cp950 ?)
CP20948  A TNS   Taiwanese PC National Standard code
CP30948  A TTEL  Taiwanese PC Telephone code
CP00857  A ASCR  IBM PC Turkish (Latin 5)
CP00920  A ISO9  ISO-8859-9 (Latin 5, Turkish)
CP01026  E EBCR  IBM MF Turkish

CP01252  A SBCS  Windows, Latin 1
CP00819  A SBCS  ISO-8859-1 (Latin 1, IBM Version 8X/9X undef'ed)
CP00912  A ISO2  ISO-8859-2 (Latin 2)
CP01047  E SBCS  IBM MF Open Systems (Latin 1)
CP65001  A UTF8  Unicode (UTF-8)
CP00916  A ISO8  ISO-8859-8 (Hebrew)
CP00813  A ISO7  ISO-8859-7 (Greek)
CP00870  E EBCE  IBM MF Multilingual (Latin 2, East European)
CP00852  A ASCE  IBM PC Multilingual (Latin 2, East European)
CP01250  A WINE  Windows, Latin 2 (East European)
CP01253  A WING  Windows, Greek
CP01254  A WINR  Windows, Turkish
CP01255  A WINH  Windows, Hebrew
CP01025  E EBCY  IBM MF Cyrillic, Multilingual
CP00866  A ASCY  IBM PC Cyrillic #2
CP00915  A ISO5  ISO-8859-5 (Cyrillic, 8-bit)
CP01251  A WINY  Windows, Cyrillic
CP01112  E EBCB  IBM MF Baltic Multilanguage
CP00921  A ISO3  ISO-8859-4 (Latin 4, Baltic Multilanguage)
CP01257  A WINB  Windows, Baltic
CP01089  A ISO6  ISO-8859-6 (Arabic)
CP01256  A WINA  Windows, Arabic
CP00838  E EBCI  IBM MF Thai
CP00874  A WINI  IBM PC Thailand
CP90278  E SBCS  IBM MF Finland, Sweden (Internal Use, zapped)
CP00914  A ISO4  ISO-8859-4 (Latin 4, Baltic Multilanguage)
CP65002  E UTFE  Unicode (UTF-EBCDIC)
CP00923  A ISOF  ISO-8859-15 (Latin 9)
CP30930  E SOSI  Japanese IBM MF Dollar-Yen exchanged CP00930
CP11026  E EBCR  IBM MF Turkish DoubleQuote-U-umlaut exchanged CP01026

```

Localizing the WebFOCUS Reporting Server Console

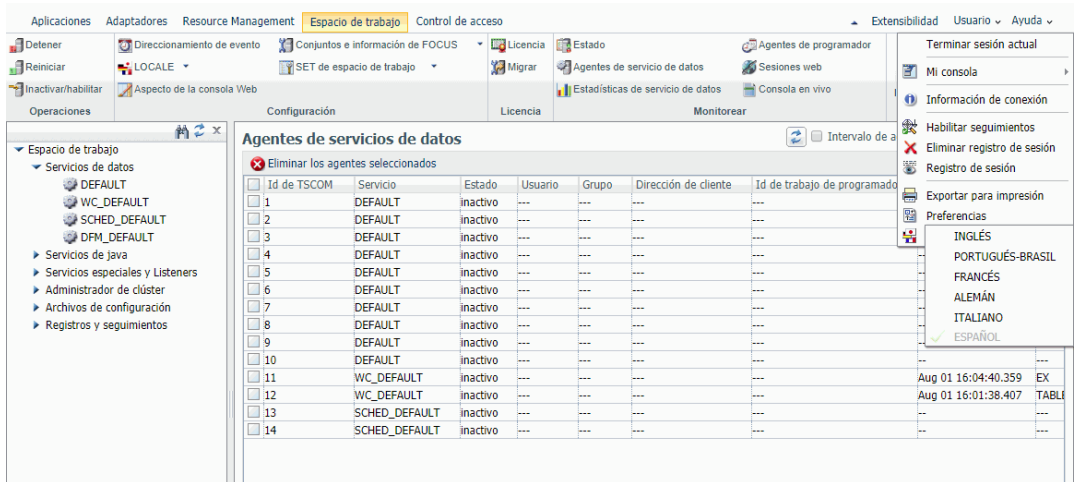
When you configure NLS on the WebFOCUS Reporting Server Console, you must select a value for the LANG attribute.

Procedure: How to Use Dynamic Language Switch on the Reporting Server Console

1. In the Reporting Server Console, on the menu bar, click *User*.
2. From the User menu, point to *Language*, and then select a language from the list.

Note that the available languages depend on the code page set in the NLS Configuration Wizard. For example, if you select a European language code page, the Western languages will be available, as well as English, whereas if you select a Japanese code page, Japanese will be as well as English, but not Western European languages. If you select a Unicode code page, all localized languages will be available.

In this example, Spanish has been chosen from the menu so the Web Console, as well as any translated error messages, are displayed in the selected language:



Procedure: How to Create a Localized Version of the WebFOCUS Reporting Server Console

When adding a localized language to the WebFOCUS Reporting Server Console, the following files must be created for each language. These files are located in the EDHOME\nls directory (for example, drive:\jbi\sr77\home\nls).

Each file contains the list of strings that are used in specific product areas.

Area	String
Web Console	wc***.lng
Synonym Editor	nt***.lng
Data Management Console	wc***.lng and dm***.lng

where:

Represents the 3-letter language abbreviation code for the new language you will be translating.

To localize the files listed in the chart, complete the following steps:

1. Identify the abbreviation code of the new language. To obtain a list of available codes, run the ? LANG SET command as a procedure. The column LANGAB (Language Abbreviation) shows the 3-letter (***) codes.
2. Copy the three component files that have the English strings and rename them using the new language code (the English versions of the files use the 'eng' language abbreviation code).
3. Translate the files with the new language abbreviation codes. To do so, open each component file in a text editor and proceed with the translation using the following guidelines:
 - ☐ Translate the strings on the right side of the "=" sign. (The left side of the = sign contains the string numbers. Do not change these values.)
 - ☐ Some strings contain multiple variables (placeholders), such as %s/%s. Be especially careful to place these strings in context within your translation.

For example, in the string:

```
4040 = Create Synonym for %s %s
```

```
%s %s
```

represents the table name for which you are creating a synonym. In some languages, the table name may be identified in the beginning or the middle of the sentence, therefore the variable must be placed accordingly.

4. Save the translated file in EDAHOME\nls directory (*drive:\ibi\srvt77\home\nls*) using an encoding system that is compatible with the default code page of the language. To check the default code page, run the ? *LANG SET* command as a procedure. The column CODEPG (Code Page) shows the code page.
5. Once the new language files exist, you can add the localized language to the dynamic language switch menu by including the ADDLANG parameter in the nlscfg.err file.

To edit the nlscfg.err file, on the Reporting Server Console menu bar click *Workspace*. In the navigation pane, expand *Configuration Files* and *Miscellaneous*, then open *LOCALE - nlscfg.err*.

The nlscfg.err file opens in the text editor. Add the following line of code:

ADDLANG=language_name

where:

language_name

Is the language abbreviation code (LANGAB) or language name (LANGNM) from the ? *LANG SET* table.

6. To test the new language, configure the WebFOCUS Reporting Server for NLS and select the language you just translated as the *language_name* value.
7. Save and restart the Server. When the Server restarts, the localized language option will be available in the User menu under Language.
8. Select the desired language and verify that all menus and options are translated and functioning.

Note: For your convenience, two reference charts that contain the output generated by the ? *LANG SET* command are included in this document. For details, see [Language and Default Code Pages for Windows, UNIX, and Linux](#) on page 74 or [Language and Default Code Pages for IBM i and z/OS](#) on page 76.

Example: Localizing the Web Console Into Finnish

For this example, assume that the server is running on Windows and the NLS setting is *CODE_PAGE=1252* and *LANG=AMENGLISH*.

Perform the following steps:

1. The *wceng.lng* file contains the original English strings for the Web Console. Copy the file and rename it to *wcfin.lng*.
2. Translate the *wcfin.lng* file into Finnish and save it with Latin 1 or ANSI encoding in the *drive:\ibi\srvt77\home\nls* directory.

- From the Reporting Server Console menu bar, click *Workspace*. In the navigation pane, expand *Configuration Files* and *Miscellaneous*, then open *LOCALE - nlscfg.err*.

The nlscfg.err file opens in the text editor. Add the following line of code:

```
ADDLANG=FINNISH
```

- Click the *Save and Restart* button for the changes to take effect.
- To apply the Finnish translation, in the Web Console, open the User menu, point to *Language*, and click *FINNISH*.

Reference: Language and Default Code Pages for Windows, UNIX, and Linux

This chart provides information generated by the ? LANG SET command from the Windows, UNIX, and Linux platforms.

Language Code	Language Name	Language Abbreviation	Language ID	Code Page
001	AMENGLISH	AME	en	1252
001	ENGLISH	ENG	en	1252
020	ARABIC	ARB	ar	1256
10351	B-PORTUGUESE	BRA	br	1252
420	CZECH	CZE	cs	1250
045	DANISH	DAN	da	1252
031	DUTCH	DUT	nl	1252
372	ESTONIAN	EST	et	1256
358	FINNISH	FIN	fi	1252
033	FRENCH	FRE	fr	1252
033	FRENCH	FRE	fc	1252
049	GERMAN	GER	de	1252
049	GERMAN	GER	at	1252

Language Code	Language Name	Language Abbreviation	Language ID	Code Page
030	GREEK	GRE	el	1253
972	HEBREW	HEB	iw	1255
036	HUNGARIAN	HUN	hu	1250
039	ITALIAN	ITA	it	1252
081	JAPANESE	JPN	ja	942
081	JAPANESE-EUC	JPE	je	10942
082	KOREAN	KOR	ko	949
371	LATVIAN	LVA	lv	1257
370	LITHUANIAN	LTU	lt	1257
047	NORWEGIAN	NOR	no	1252
048	POLISH	POL	pl	1250
351	PORTUGUESE	POR	pt	1252
040	ROMANIAN	ROM	ro	1250
007	RUSSIAN	RUS	ru	1251
085	S-CHINESE	PRC	zh	946
034	SPANISH	SPA	es	1252
046	SWEDISH	SWE	sv	1252
086	T-CHINESE	ROC	tw	10948
066	THAI	THA	th	874
090	TURKISH	TUR	tr	1254
044	UKENGLISH	UKE	uk	1252

Language Code	Language Name	Language Abbreviation	Language ID	Code Page
800	UNICODE	UCS	uc	65001

Reference: Language and Default Code Pages for IBM i and z/OS

This chart provides information generated by the ? LANG SET command from the IBM i and z/OS platforms.

Language Code	Language Name	Language Abbreviation	Language ID	Code Page
001	AMENGLISH	AME	en	37
001	ENGLISH	ENG	en	37
020	ARABIC	ARB	ar	420
10351	B-PORTUGUESE	BRA	br	37
420	CZECH	CZE	cz	870
045	DANISH	DAN	da	277
031	DUTCH	DUT	nl	37
372	ESTONIAN	EST	et	1112
358	FINNISH	FIN	fi	278
033	FRENCH	FRE	fr	297
033	FRENCH	FRE	fc	297
049	GERMAN	GER	de	273
049	GERMAN-500	GE5	at	500
030	GREEK	GRE	el	875
972	HEBREW	HEB	iw	424

Language Code	Language Name	Language Abbreviation	Language ID	Code Page
036	HUNGARIAN	HUN	hu	870
039	ITALIAN	ITA	it	280
081	JAPANESE	JPN	ja	939
10081	JAPANESE-930	JPK	jk	930
082	KOREAN	KOR	ko	933
371	LATVIAN	LVA	lv	1112
370	LITHUANIAN	LTU	lt	1112
047	NORWEGIAN	NOR	no	277
048	POLISH	POL	pl	870
351	PORTUGUESE	POR	pt	37
040	ROMANIAN	ROM	ro	870
007	RUSSIAN	RUS	ru	1025
085	S-CHINESE	PRC	zh	935
042	SLOVAK	SVK	sk	870
034	SPANISH	SPA	es	284
046	SWEDISH	SWE	sv	278
086	T-CHINESE	ROC	tw	937
066	THAI	THA	th	838
090	TURKISH	TUR	tr	1026
044	UKENGLISH	UKE	uk	285
800	UNICODE	UCS	uc	65002

Configuring National Language Support for the Data Management Console

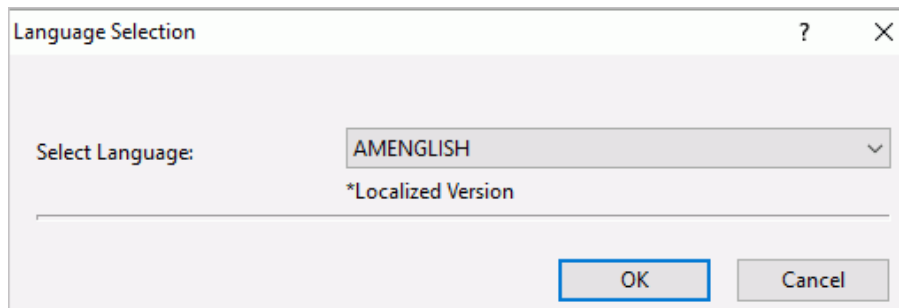
In addition to the WebFOCUS Reporting Server Console, you can manage your Reporting Server from the Data Management Console, or DMC. Like the Reporting Server Console, the DMC allows you to manage adapters and create and edit synonyms. In addition, it allows you to create data flows and process flows to extract data from data sources and transform that data before it is loaded to a data target.

You can configure NLS directly from the DMC. National Language Support enables the connector or server to translate both double-byte and single-byte National Character sets.

Procedure: How to Configure NLS in the Data Management Console

1. On the Home tab, in the Tools group, click *Language Selection*.

The Language Selection dialog box opens, as shown in the following image.



The Language Selection dialog box has the following field:

Select Language

Controls the default value of the code page and the language of server error messages if they are available for translation. The default value is AMENGLISH (American English) on all platforms. Localized versions are noted by an asterisk (*).

2. Select a Language from the drop-down menu. (Note that localized versions are noted with an asterisk (*).)
3. Click *OK*.

Configuring a WebFOCUS Client for NLS

This topic describes the steps for configuring a WebFOCUS Client for National Language Support (NLS) on Windows, UNIX, or IBM i, and z/OS using the supplied web-based consoles.

Configuration through the web-based consoles provides standard character transcoding, monocasing, and sorting and is sufficient for most enterprises.

In this chapter:

- ❑ [Configuring a WebFOCUS Client for NLS](#)
 - ❑ [Displaying National Characters on Sun Solaris](#)
 - ❑ [Adding NLS Information to the Communications Configuration File](#)
 - ❑ [ReportCaster Distribution Server Support for UTF-8 on Windows](#)
-

Configuring a WebFOCUS Client for NLS

To configure a WebFOCUS Client for National Language Support (NLS), sign in to the WebFOCUS Administration Console and select a code page.

By default, the WebFOCUS Client uses the following code pages if installed on a non-English operating system. (For Western European operating systems the default can be used. For other languages it is necessary to reconfigure):

- ❑ 1252 on Windows, UNIX, and Linux
- ❑ 37 on IBM i and z/OS

Procedure: How to Configure a WebFOCUS Client for NLS (Windows, UNIX, Linux, IBM i, z/OS (HFS))

1. Open the WebFOCUS Administration Console with administrative privileges.

In a browser, enter the following address:

http://host:port/ibi_apps/admin

where:

host

Is the host name or IP address of the machine running the WebFOCUS Client.

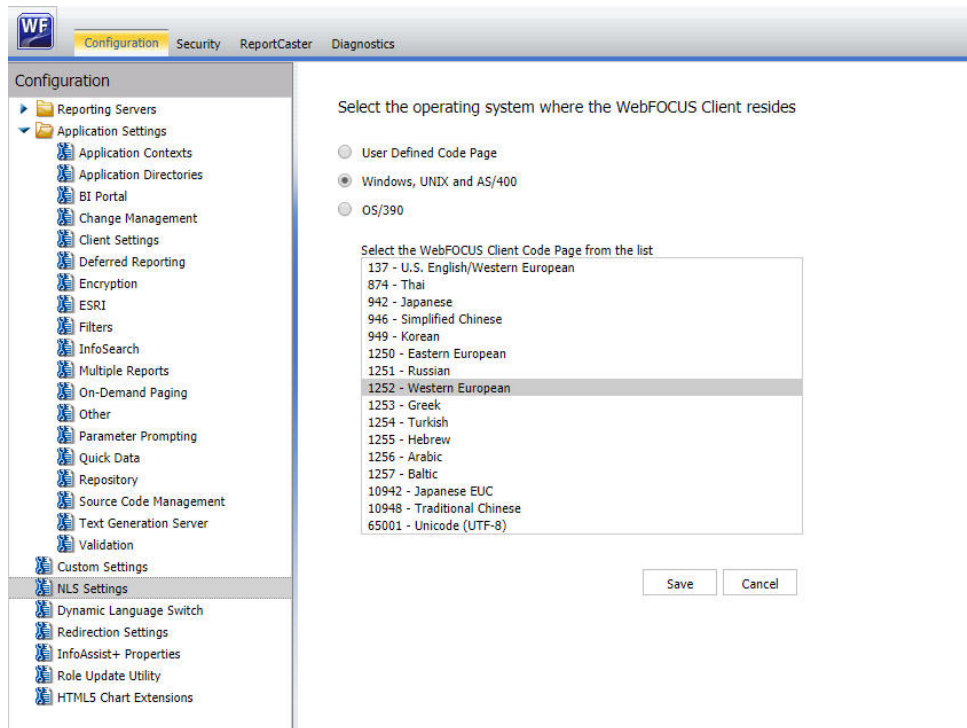
port

Is the port number on which the server is listening.

The WebFOCUS Sign In page opens. Sign in as an administrator.

If you are working within App Studio, open the *WebFOCUS Administration* menu and click *WebFOCUS Administration Console*.

2. In the Administration Console navigation pane, click *NLS Settings*. The NLS Settings page is shown in the following image.



3. Select the operating system on which the WebFOCUS Client is running.
The code pages for the selected operating system are displayed in the list.
4. From the list, select a code page that configures the client for the correct display of report output in the browser.

The language selected for the client usually corresponds to the language selected for the server from the WebFOCUS Reporting Server Console. The code page for the client must be the ANSI equivalent of the code page for the server you are reporting from.

If the code page chosen from the WebFOCUS Reporting Server Console does not appear in the list box on the WebFOCUS Administration Console, select *User Defined Code Page* and type the code page number in the text box.

You would do this, for example, if the server adds support for a new code page that is not yet reflected in the client software.

5. Click **Save** to store the settings. The client configuration file is updated with the selected code page.

Reference: Using JDK 1.7 on Windows

During a WebFOCUS installation in Windows, the default WebFOCUS language is set to the system's locale, .DEFAULT user ID, even though the current user ID is set to a different language. This is a JDK limitation.

WebFOCUS depends on the JDK file.encoding value to appropriately read and write data, and display NLS characters in the tools.

To determine the value for file.encoding, in the Administration Console, access click the *Diagnostics* tab and click *JVM Property Info*. The value for file.encoding is available in the System Properties list.

Displaying National Characters on Sun Solaris

Solaris has changed its default locale system of Java encoding from ISO-8859-1 to UTF8. As a result, national characters are displayed as two question marks (??).

To correct the problem, you must include the ISO encoding parameter in the startup shell of the web server and servlet engine.

```
export LANG=languagecode_COUNTRYCODE.ISOXXXX-X
```

where:

languagecode

Is the 2-letter ISO language code. See <http://www.w3.org/WAI/ER/IG/ert/iso639.htm>.

COUNTRYCODE

Is the 2-letter ISO country code. See http://www.iso.org/iso/prods-services/iso3166ma/02iso-3166-code-lists/country_names_and_code_elements.

ISOXXXX-X

Is one of the following:

- ☐ ISO8859-1 (Latin 1)
- ☐ ISO8859-2 (Latin 2)
- ☐ ISO8859-3 (Latin 3)
- ☐ ISO8859-4 (Baltic)
- ☐ ISO8859-5 (Cyrillic)
- ☐ ISO8859-6 (Arabic)
- ☐ ISO8859-7 (Greek)
- ☐ ISO8859-8 (Hebrew)
- ☐ ISO8859-9 (Turkish)
- ☐ ISO8859-15 (Latin 9)

Example for English:

```
export LANG=en_US.ISO8859-1
```

Example for German:

```
export LANG=de_DE.ISO8859-1
```

Adding NLS Information to the Communications Configuration File

You can add code page information to the communications block of the server or client communications configuration file for documentation purposes, so you know which code page applies to which server or client node.

On Windows, UNIX, Linux, and IBM i. The communications configuration file, `odin.cfg`, is located in the `etc` directory under the configuration directory. For example, on Windows, the server communications configuration file would be located in `drive:\ibi\srv77\wfs\etc`, and the client communications configuration file would be located in `drive:\ibi\WebFOCUS82\client\wfc\etc`.

You can also access the server communications configuration file through the Web Console. In the Workspace tab, in the navigation pane, expand *Configuration Files* and open *Communication - odin.cfg*.

On z/OS. The client communications file is member `ODIN` in `qualif.EDACTL.DATA`.

Syntax: How to Add NLS Information to the Communications Configuration File

`CODE_PAGE =`

Is the code page.

Example: Adding NLS Information to the Communications Configuration File (z/OS)

The following is an example of documentation for the code page setting for the server in the client communications configuration file, `ODIN`.

```
.
.
.
BEGIN
  PROTOCOL = TCP
  HOST = IBIMVS
  SERVICE = 2620
  CLASS = CLIENT
  CODE_PAGE = 500
END
```

ReportCaster Distribution Server Support for UTF-8 on Windows

Although historically the ReportCaster Distribution Server had its own NLS settings, as of WebFOCUS Release 8.2, ReportCaster uses the NLS settings configured for the WebFOCUS Client.

If your WebFOCUS Client is configured to use a Unicode UTF-8 code page, such as 65001, you may encounter issues in which the ReportCaster distribution of certain output times does not apply the correct code page, resulting in instances of incorrect character output.

To resolve this, you can add `file.encoding` and `user.language` properties to the `schbkr.bat` file in your ReportCaster installation and to the Java Options parameter for your WebFOCUS installation in the Registry.

These processes are described in the following procedure.

Procedure: How to Configure UTF-8 for the ReportCaster Distribution Server

If you encounter text encoding issues in your ReportCaster distribution, you can add the `file.encoding` and `user.language` properties to the `schbkr.bat` file in your installation as a step to resolving the inconsistency.

1. In Windows Explorer, navigate to the `bin` folder in your ReportCaster installation, for example, `drive:/ibi/WebFOCUSnn/ReportCaster/bin`, where *nn* is the version of WebFOCUS that you have installed.
2. Right-click `schbkr.bat` and click *Edit* to open the file in a text editor.
3. Add the following text to the file to specify the UTF-8 configuration for the ReportCaster Distribution Server:

```
-Dfile.encoding=UTF8
```

4. Optionally, to specify a language other than English, add the following text after `-Dfile.encoding=UTF8`.

```
-Duser.language=ln
```

where:

ln

Is a two-letter language ID for the selected language, such as *fr* for French. The language ID should be typed in lowercase. For a list of available options, see [Language and Default Code Pages for Windows, UNIX, and Linux](#) on page 74.

Once finished, save and close `schbkr.bat`.

5. Open the Windows Registry Editor.
6. In the navigation pane on the left, expand `WOW6432Node`, *Information Builders*, *ReportCaster*, *WFnn* (where *nn* represents your installation of WebFOCUS), *Parameters*, and *Java*.
7. Double-click *Options*.

The Edit Multi-String dialog box opens.

8. On a new line, type the following text to specify the UTF-8 configuration for the ReportCaster Distribution Server:

```
-Dfile.encoding=UTF8
```

9. Optionally, to specify a language other than English, type the following text on a new line after `-Dfile.encoding=UTF8`.

```
-Duser.language=ln
```

where:

ln

Is a two-letter language ID for the selected language, such as *fr* for French. The language ID should be typed in lowercase. For a list of available options, see [Language and Default Code Pages for Windows, UNIX, and Linux](#) on page 74.

10. Click *OK* to submit your changes.

These changes will be applied the next time you start WebFOCUS.

Configuring WebFOCUS App Studio for NLS

This section describes the steps for configuring WebFOCUS App Studio for National Language Support (NLS). It identifies the architectural components of App Studio to help you understand the NLS configuration requirements.

To help you configure NLS at your enterprise, this section also provides a configuration summary.

You need to perform only two easy steps to configure App Studio for NLS.

In this chapter:

- ❑ [App Studio Architecture](#)
 - ❑ [App Studio Configuration Steps](#)
 - ❑ [App Studio Configuration Summary](#)
-

App Studio Architecture

The architecture of WebFOCUS App Studio makes configuring NLS easy and transparent. Its components include:

- ❑ **WebFOCUS Reporting Server.** Handles the selection of records from data sources and the generation of report output from those data sources. The data access component can handle data in ANSI (Windows) format. For information on configuring the WebFOCUS Reporting Server for NLS, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59.
- ❑ **WebFOCUS Client.** Resides on the web server and application server. It connects the WebFOCUS Reporting Server to the web through the CGI, ISAPI, or Java servlets. You must set the code page for the client, using the WebFOCUS Administration Console, to tell the Reporting Server how to format the data for the WebFOCUS Client platform. The WebFOCUS Client code page setting controls transcoding of data when a report is run. For information on configuring the WebFOCUS Client for NLS, see [Configuring a WebFOCUS Client for NLS](#) on page 79.

- ❑ **App Studio.** Consists of graphical development and code generation tools that can write or read procedures and synonyms in ANSI format.

App Studio Configuration Steps

Step 1. Verify That the Server Is Configured for the Code Page of the Data Source

App Studio requires configuration of the WebFOCUS Reporting Server for NLS through the WebFOCUS Reporting Server Console. Verify that the WebFOCUS Reporting Server has been configured for the code page of the data source.

The code page may be an ANSI (Windows) or EBCDIC (mainframe) code page. In general, most Relational Database Management System (RDBMS) tables are stored with ANSI code pages.

Step 2. Verify That the Client is Configured for the Correct Display of Report Output

App Studio requires configuration of the WebFOCUS Client for NLS through the Administration Console. Verify that the WebFOCUS Client has been configured for the correct display of report output in the browser.

Instructions for configuring the WebFOCUS Client for NLS can be found in [Configuring a WebFOCUS Client for NLS](#) on page 79.

Step 3. Configure App Studio for NLS

You must ensure that all application components are stored with a consistent encoding scheme. For example, you must configure App Studio appropriately for the code page of the procedure and synonym that will be processed so that the WebFOCUS Reporting Server knows how to format the data for display.

App Studio uses ANSI code page UTF-16 by default to store procedures and Master Files. The WebFOCUS Reporting Server also reads data sources in ANSI format by default.

You can change the language in App Studio using Dynamic Language Switch in the App Studio Options dialog box. To access the App Studio Options dialog box, open the Application menu and click *Options*. In the General section, you can select a language from the Language drop-down menu.

Example: Processing a Procedure With National Characters

A procedure stored by App Studio can pass text as lines of headings and as record selection values to the data access component. A Master File can pass column titles or record selection criteria in an ACCEPT attribute. Consider the following procedure.


```
TABLE FILE EMPHR
HEADING CENTER
"Société Française Noël en Fête"
"Rapport des Ressources Humaine"
SUM EMP_COUNT
RESIGN_COUNT
FIRE_COUNT
COMPUTE TotalPersonnel/I3 = EMP_COUNT + RESIGN_COUNT + FIRE_COUNT;
WHERE PLANTLNG EQ 'Montréal'
END
```

The procedure contains national characters embedded in the heading as well as in the WHERE phrase that controls selection of records from the EMPHR data source.

App Studio Configuration Summary

This topic summarizes how to store application resources and configure NLS on the server, the client, and App Studio.

For details on setting the code page on the WebFOCUS Reporting Server and WebFOCUS Client, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59 and [Configuring a WebFOCUS Client for NLS](#) on page 79.

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

If the target data sources are ANSI code page-based, the following configuration is necessary:

- ❑ **Data, procedures, and synonyms.** All files accessed by server and client must be stored with an ANSI code page. You can place the code page parameter (CODE_PAGE) in the synonym for documentation. For more information, see [Adding the Code Page Parameter to a Master File](#) on page 158.
- ❑ **WebFOCUS Reporting Server Configuration.** The server must be configured for an ANSI code page, such as 1252 for Western European languages.
- ❑ **WebFOCUS Client Configuration.** The WebFOCUS Client must be configured for an ANSI code page to display report output correctly during a browser session.
- ❑ **App Studio Configuration.** App Studio must read and store procedures and synonyms in ANSI format. You do not need to do anything to configure App Studio.

Configuring for Double-Byte Character Languages

There is no distinction between ANSI (Windows) format for the double-byte character sets that represent Asian languages. Configuration should be as follows:

- ❑ **WebFOCUS Reporting Server and WebFOCUS Client Configuration.** The server must be configured for the appropriate code page for the target language, such as Chinese or Japanese.
- ❑ **App Studio Configuration.** App Studio must be configured to read and store procedures and synonyms in ANSI format. You do not need to do anything to configure App Studio.

Unicode and the WebFOCUS Reporting Server

The introduction of Unicode into Information Builders core text-handling facilities has given its products the flexibility to support true multilingual text. Using Unicode formatted data, WebFOCUS can produce a report containing data in any combination of languages (for example, Japanese and French).

Information Builders supports UTF-8 Unicode on the WebFOCUS Reporting Server. UTF-8 Unicode is not a code page, but it is treated as such in Information Builders product architecture. Information Builders has created a code page of UTF-8 values for all supported scripts.

In this chapter:

- ❑ [Unicode Support on the WebFOCUS Reporting Server](#)
 - ❑ [Accessing Unicode Data](#)
 - ❑ [Selecting, Reformatting, and Manipulating Characters](#)
 - ❑ [Sort Order Under Unicode](#)
 - ❑ [Create a Unicode Environment Without Using the Tomcat Administration Tool](#)
 - ❑ [Unicode PDF Output](#)
-

Unicode Support on the WebFOCUS Reporting Server

The WebFOCUS Reporting Server supports a Unicode Transformation Format (UTF) called UTF-8 in ASCII environments, and UTF-EBCDIC in EBCDIC environments:

- ❑ **For ASCII**, the UTF-8 encoding standard assigns each character of each language a code that can be from one to three bytes long. The codes assigned to European characters are one or two bytes long, Middle Eastern characters are two bytes long, and those assigned to Asian characters are three bytes long. This standard is compatible with ASCII format because the first 128 UTF-8 codes have the same one-byte representation as the corresponding ASCII codes.
- ❑ **For EBCDIC**, the UTF-EBCDIC encoding standard assigns each character a code that can be from one to four bytes long. EBCDIC characters, including C1 control characters, have the same one-byte representation in UTF-EBCDIC.

In non-Unicode, single-byte encoding standards, such as ASCII, each character is assigned a code that is one byte long, limiting the number of characters that can be represented by the standard. When using those standards, it became common to equate a character with a byte of storage. If you had a string of 10 characters, the amount of storage needed was 10 bytes, and many character manipulation routines expected character string lengths to be specified as a number of bytes.

With Unicode encoding, bytes and characters are no longer equated. Characters are represented internally by a varying number of bytes, depending on the character. If you configure the server for Unicode, you define the length of strings and alphanumeric fields in terms of characters, not bytes. This simplifies specifying string and field lengths. Each character is represented internally by up to three bytes, and the server automatically adjusts for the actual storage length. In reports, each character displays in a report column using one space, regardless of how many bytes it takes up in memory. This character-based processing mode employed for Unicode environments is called *character semantics*. The non-Unicode mode is called *byte semantics*.

Procedures will continue to work when deployed in a Unicode environment, without adjustment, in most cases.

The main benefit of the new system is the ability to have multiple languages (both European and Asian) in the following WebFOCUS and Dialogue Manager objects:

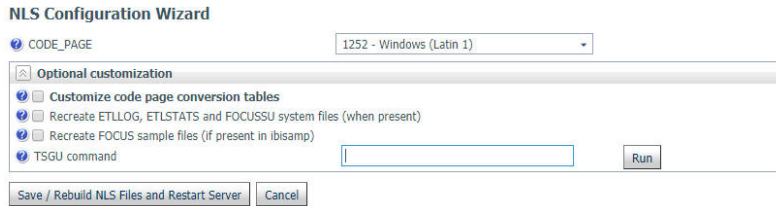
- ☐ Titles, descriptions, and names in synonyms.
- ☐ Headings and prompts in procedures.
- ☐ Data for all supported adapters (for example, SAP BW, SAP R/3-ECC, Oracle, Db2, Sybase ASE, Teradata, MySQL, Web Services, Fixed files). For more information, see [Accessing Unicode Data](#) on page 93.

***Procedure:* How to Configure the Server for Unicode**

To configure the server for UTF-8 or UTF-EBCDIC character encoding:

1. In the WebFOCUS Reporting Server Console, select *Workspace* from the menu bar.
The Data Services Agents page opens.
2. On the Workspace tab, in the Settings group, click *LOCALE* and then click *Configuration Wizard*.

The NLS Configuration Wizard pane opens, as shown in the following image.



3. Select *65001 - Unicode (UTF-8)* or *65002 - Unicode (UTF-EBCDIC)* from the CODE_PAGE drop-down list.
4. Click *Save and Restart*.

The server will be configured for Unicode.

Accessing Unicode Data

Adapters for the following types of data sources support Unicode:

- ☐ [Unicode Considerations for Db2](#) on page 94
- ☐ [Unicode Considerations for Fixed-Format Sequential Files](#) on page 94
- ☐ [Unicode Considerations for Microsoft SQL Server](#) on page 95
- ☐ [Unicode Considerations for MySQL](#) on page 96
- ☐ [Unicode Considerations for Oracle](#) on page 96
- ☐ [Unicode Considerations for SAP BW and SAP R/3-ECC](#) on page 97
- ☐ [Unicode Considerations for Sybase ASE](#) on page 97
- ☐ [Unicode Considerations for Teradata \(CLI\)](#) on page 98
- ☐ Web Services
- ☐ XBRL
- ☐ XML

For information about all adapters, see the *Adapter Administration* technical content.

Relational adapters in a Unicode environment assume that the DBMS returns character data to the server already converted to Unicode. The relational adapters convert data to the correct DBMS API when writing to a relational data source (for example, Oracle to UTF-8, Microsoft SQL Server to UTF-16, and Db2 on MVS to UTF-EBCDIC).

XML-based adapters obtain the code page from the XML declaration of the processed XML document.

The Adapter for Web Services generates SOAP requests using the UTF-8 code page.

Reference: Unicode Considerations for Db2

Information Builders supports Db2 databases, version 8 and higher. To prepare the Db2 environment for Unicode on:

- ❑ **Windows.** The database must have been created with the option CODESET UTF-8, and you must add the following variable to the environment using Windows or in the edastart file:

```
DB2CODEPAGE=1208
```

- ❑ **UNIX, Linux.** The database must have been created with the option CODESET UTF-8, and you must set the LANG and NLS_LANG environment variables in the edastart file or in a separate shell file.

For example, for American English, you would export the following variables:

```
export LANG=EN_US.UTF-8
```

- ❑ **z/OS.** The database must have been created with the CCSID UNICODE option, and you must ensure that the DSNAOINI environment variable points to a configuration file containing the following specification:

```
CURRENTAPPENSCH=UNICODE
```

The adapter supports Unicode only with the CLI interface.

In a Unicode environment, the Adapter for Db2 requires a BIND command for PREPARE/EXECUTE logic using parameter markers.

For information about data type support, see [Relational Adapter Data Type Support for Unicode](#) on page 98.

Reference: Unicode Considerations for Fixed-Format Sequential Files

When retrieving a fixed-format sequential file, the server attempts to determine the code page the file was meant to be retrieved with by checking the Master File's CODEPAGE attribute. If the Master File does not contain the CODEPAGE attribute, the server uses the value specified by the APP PROPERTY CODEPAGE command, if one was issued. If a code page was not specified by the attribute nor by the command, the server code page is used to read the file.

If you use the Data Management Console to generate a data flow that creates a fixed-format sequential file, you can specify a code page in the Properties panel for the target object. DataMigrator will then create the fixed-format file in a way that can be read by the server when that server has been configured for the specified code page.

In a Unicode configuration, HOLD files in BINARY and ALPHA formats are created using UTF-8 conversion, which assigns each character three bytes of storage in ASCII environments or four bytes in EBCDIC environments. Fields defined in the Master File using the data type A in both the USAGE and ACTUAL attributes are described in terms of characters. Fields defined using any other combination of USAGE and ACTUAL attribute values are described in terms of bytes.

To force a field in a fixed-format sequential file to be described in terms of bytes, add B to the end of the field's ACTUAL attribute. For example, to specify that a field is stored in 10 bytes, you would specify:

```
ACTUAL=A10B
APP PROPERTY appname CODEPAGE pagenum
```

The adapter will then read the specified number of bytes from the record and convert their contents to the number of characters specified by the file code page.

Regardless of how much storage a character occupies, it occupies only one space on a report, as always.

To set an application's code page, issue the following command from the application's profile or a server profile:

```
APP PROPERTY appname CODEPAGE pagenum
```

where:

appname

Is the name of the application.

pagenum

Is the number of the code page that the server will use to read a fixed-format sequential file in the named application.

Reference: Unicode Considerations for Microsoft SQL Server

The adapter, using the OLE DB interface, supports Unicode data stored in NCHAR and NVARCHAR fields (where N stands for national). N columns can support data of any language or combination of languages.

For information about data type support, see [Relational Adapter Data Type Support for Unicode](#) on page 98.

Reference: Unicode Considerations for MySQL

The Adapter for MySQL is implemented using JDBC. This implementation supports Unicode data stored in character fields with CHARACTER SET set to UTF-8.

You must set the LANG environment variable in the edastart file or in a separate shell file before you start the server. For example, for American English, you would export the following variable:

```
export LANG=EN_US.UTF-8
```

or

```
export LANG=en_US.utf-8
```

For information about data type support, see [Relational Adapter Data Type Support for Unicode](#) on page 98.

Reference: Unicode Considerations for Oracle

The adapter supports Unicode data in Oracle release 10g or higher databases that have been configured with the NLS_CHARACTERSET parameter set to either UTF8 or AL32UTF8. However, AL32UTF8 is preferred. You must set the NLS_LANG environment variable in the edastart file, in a separate shell file, in a database profile, or in a user profile.

Set NLS_LANG using the following syntax:

```
NLS_LANG = language_territory.characterset
```

For example, for American English:

```
NLS_LANG=American_America.AL32UTF8
```

where:

language

Is the selected language.

territory

Is the name of the country associated with the selected language.

characteraset

Is the value of the NLS_CHARACTERSET variable that is set in the Oracle database. For Unicode, this can be AL32UTF8 or UTF8. AL32UTF8 is preferred.

For example, for American English UTF-8, you would use the following setting:

```
NLS_LANG=American_America.AL32UTF8
```

For information about data type support, see [Relational Adapter Data Type Support for Unicode](#) on page 98.

Reference: Unicode Considerations for SAP BW and SAP R/3-ECC

SAP uses UTF-16 encoding in its Unicode system. The server uses UTF-8 and handles all conversions between the two encoding schemes.

NLS settings for the Reporting Server should be configured in such a way that the Application Server code page can handle the list of chosen languages. For example, ISO 8859-1 can accommodate most Western European languages. The 8859 family can handle character specifics with the lower set almost being mapped to US ASCII. Therefore, with 8859-1 one could request English, German, French, and Spanish. When a character set requires a code page that takes more than one byte per character (for example, many Asian languages), the only choice for the server is 65001 (UTF-8).

The adapters provide access to Unicode SAP BW and SAP ECC systems, respectively. This extends support of data and metadata in multiple languages to the server, consistent with support by the SAP server. A synonym can be created using one or more languages. Those languages will be used to create titles and descriptions.

- ☐ For SAP BW, the userid and password credentials must be able to connect using the enumeration of desired languages.
- ☐ For SAP R/3-ECC, the server sign-in language is used to retrieve all languages.

Reference: Unicode Considerations for Sybase ASE

The adapter supports Unicode data in Sybase ASE version 15.0 and higher databases that have been created with the CHARACTER SET option set to UTF-8. You must set the LANG variable in the edastart file or in a separate shell file before starting the server.

For example, for American English, you would export the following variables:

```
export LANG=EN_US.UTF-8
```

For information about data type support, see [Relational Adapter Data Type Support for Unicode](#) on page 98.

Reference: Unicode Considerations for Teradata (CLI)

The Adapter for Teradata (CLI) supports Unicode UTF-8 format if:

- ❑ The Teradata CLI client components are part of release TTU8.0 or higher.
- ❑ The Teradata database is release V2R6.0 or higher and appropriate language support was enabled during the sysinit process.

Contact your database administrator (DBA) to determine whether international language support has been enabled in your Teradata system and/or consult the Teradata documentation for details about International Character Set support.

Note that, at the present time, when Unicode is enabled, the length of a Teradata Column Name and/or TITLE cannot exceed 21 characters (bytes).

For information about data type support, see [Relational Adapter Data Type Support for Unicode](#) on page 98.

Reference: Relational Adapter Data Type Support for Unicode

In Unicode databases the information in CHAR(n) columns is stored in a UTF-8 encoding scheme. Most RDBMS Unicode columns of CHAR type specify length in bytes, not characters; the B-modifier in the Actual format denotes that a character column with a fixed byte length might contain a varying number of UTF-8 characters. This is reflected in the AnV Usage format, as shown in the following table.

DBMS	Column Type	Usage	Actual *
Db2	CHAR(n)	AnV	AnB
	GRAPHIC(n)	An	An
	VARCHAR(n)	AnV	AnVB
	VARGRAPHIC(n)	AnV	AnV

DBMS	Column Type	Usage	Actual *
Microsoft SQL Server	CHAR(<i>n</i>) single byte code page	An	An
	CHAR(<i>n</i>) double byte code page	AnV	AnV
	NCHAR(<i>n</i>)	An	An
	VARCHAR(<i>n</i>)	AnV	AnV
	NVARCHAR(<i>n</i>)	AnV	AnV
MySQL	CHAR(<i>n</i>)	An	An
	VARCHAR (<i>n</i>)	AnV	AnV
Oracle	CHAR(<i>n</i> CHAR)	An	An
	CHAR(<i>n</i> BYTE)	AnV	AnB
	NCHAR(<i>n</i>)	An	An
	VARCHAR(<i>n</i> CHAR)	AnV	AnV
	VARCHAR(<i>n</i> BYTE)	AnV	AnVB
	NVARCHAR(<i>n</i>)	AnV	AnV
Sybase ASE	CHAR(<i>n</i>)	An	AnB
	UNICHAR(<i>n</i>)	An	An
	VARCHAR(<i>n</i>)	AnV	AnVB
	UNIVARCHAR(<i>n</i>)	AnV	AnV
Teradata761	CHAR(<i>n</i>)	An	An
	VARCHAR (<i>n</i>)	AnV	AnV

Note that on EBCDIC platforms the ACTUAL size for a B-suffixed format is increased 1.5 times to accommodate the expansion when converting from UTF-8 to UTF-EBCDIC. For example, on MVS the synonym created for a Db2 CHAR(10) column contains the following: USAGE=A10, ACTUAL=A15B.

Selecting, Reformatting, and Manipulating Characters

In character semantics mode, selection tests against a mask are automatically adjusted to work with characters rather than bytes. Formats assigned by reformatting a field in a request or by defining a temporary field are interpreted in terms of characters. Character functions interpret all lengths in terms of characters, as well.

Example: Defining a Virtual Field

Consider the following DEFINE in the Master File for the EMPLOYEE data source:

```
DEFINE FIRST_ABBREV/A5 WITH FIRST_NAME = EDIT(FIRST_NAME, '99999$$$$$');$
```

In character semantics mode, format A5 is interpreted as five characters (up to 15 bytes on ASCII platforms, up to 20 bytes on EBCDIC platforms), and the comparison is performed based on this number of bytes. In byte semantics mode, format A5 is interpreted as five bytes, and the comparison is performed based on five bytes. In either case, the correct characters are compared and extracted.

Example: Reformatting a Field

Consider the following PRINT command:

```
PRINT FIELD1/A10
```

In character semantics mode, format A10 is interpreted as 10 characters (up to 30 bytes), meaning that up to 30 bytes must be retrieved when this field is referenced. In byte semantics mode, format A10 means that 10 bytes will be retrieved. In either case, the field displays as 10 characters that take up 10 spaces on the report output.

Reference: Character Functions That Support Character Semantics

In character semantics mode, all character manipulation functions interpret lengths in terms of characters. The following functions operate on alphanumeric strings in character semantics mode when Unicode is configured:

☐ String manipulation and extraction functions.

GETTOK, OVLAY, PARAG, REVERSE, SQUEEZ, STRIP, SUBSTR, SUBSTV, TRIM, TRIMV

☐ Justification functions.

CTRFLD, LJUST, RJUST

☐ **Length and position functions.**

ARGLEN, LENV, POSIT, POSITV

☐ **Format conversion functions.**

EDIT

☐ **Decoding, comparison, and editing functions.**

CHKFMT, EDIT, DECODE, SOUNDEX

☐ **String replacement functions.**

CTRAN, HEXBYT, BYTVAL (see notes below), STRREP

☐ **Case translation functions.**

LCWORD, LOCASE, LOCASV, UPCASE, UPCASV

Note: The HEXBYT, BYTVAL, and CTRAN functions have been extended to handle multibyte characters in Unicode configurations. These functions use or produce numeric values to represent characters. In Unicode configurations, they use or produce values in the range:

☐ 0 to 255 for 1-byte characters

☐ 256 to 65535 for 2-byte characters

☐ 65536 to 16777215 for 3-byte characters

☐ 16777216 to 4294967295 for 4-byte characters (primarily for EBCDIC)

To find the numeric value corresponding to a given character, find its hexadecimal code and convert to decimal with a hex calculator, such as the Windows Calculator program. Make sure to use the UTF-8 or UTF-EBCDIC code, not the Unicode code point, which would be the UTF-16 value.

For example, assume you would like to create a variable of format A1 containing the euro sign. The euro sign in UTF-8 is, in hex, E282AC. Converting this to decimal gives 14849492. You can then use the HEXBYT function to convert a decimal value to the euro character. Thus, a DEFINE or COMPUTE to generate a euro symbol would be:

```
EUROSIGN/A1 = HEXBYT(14849492, 'A1');
```

For more information on the HEXBYT function, see [HEXBYT: Converting a Decimal Integer to a Character](#) on page 189.

If you are creating a FOCEXEC with a UTF-8 compliant editor, then instead of using the Windows calculator to find the decimal value, you can also get the value of the euro sign (€) using the BYTVAL function:

```
EUROVAL/I8 = BYTVAL('€', 'I8');
```

For more information on the BYTVAL function, see [BYTVAL: Translating a Character to Decimal](#) on page 164.

You can have this value input directly into the CTRAN function. In the example below, the decimal value generated by the BYTVAL function is stored as EUROVAL. EUROVAL is then referenced by the CTRAN function.

```
EUROVAL/I8 = BYTVAL('€', 'I8');  
NEWFLD/A40 = CTRAN(40, OLDFLD, EUROVAL, 49827, 'A40');
```

The CTRAN function replaces all occurrences of a character in a string with another character, given the decimal values that represent the hexadecimal codes for the two characters. Traditionally, this technique was used to replace characters that were difficult to input directly. However, decimal values of characters can be complicated to determine. Therefore, if you want to replace characters or character strings that you can input directly using a UTF-compliant text editor, it may be easier use the STRREP string replacement function.

The following translates all of the euro signs (€) in a 40-character UTF-8 field to pound sterling signs (£ = 49827):

```
NEWFLD/A40 = CTRAN(40, OLDFLD, EUROVAL, 49827, 'A40');
```

For more information on the CTRAN function, see [CTRAN: Translating One Character to Another](#) on page 167.

Alternatively, you could use the following STRREP function to perform the translation. This removes the step of determining the decimal values for each character, but requires you to be able to enter each character on your platform, in this case the euro character (€) and the pound sterling character (£).

```
NEWFLD/A40 = STRREP(40, OLDFLD, 1, '€', 1, '£', 40, NEWFLD);
```

For more information on the STRREP function, see [STRREP: Replacing Character Strings](#) on page 215.

Sort Order Under Unicode

Sort order is based on the binary values assigned to the characters. When the server is configured for Unicode, the sort order is based on the Unicode encoding standard. If ascending values of the codes correspond to the alphabetical order of the letters in the language being used, a report can be sorted in alphabetical order. This is entirely dependent on the encoding standard and its mapping of codes to letters. In many, but not all cases, the encoding standard assigns codes in the alphabetical order of each language.

For example, Ukrainian added a new letter (Cyrillic capital letter ghe with upturn - І) to its alphabet after the UTF-8 coding specification had already been set. This letter was not assigned a code that sorts it alphabetically, either in Unicode or in code page 1251 (used for Ukrainian). It sorts differently and incorrectly using either encoding scheme.

- ❑ With code page 1251, this letter sorts as the first letter on the report output.
- ❑ With UTF-8, this letter sorts as the last letter on the report output.

To determine whether a language sorts alphabetically, you can examine the hexadecimal codes assigned to its letters on the code page you are using and check whether ascending hexadecimal codes match the alphabetical order.

Create a Unicode Environment Without Using the Tomcat Administration Tool

1. Edit the *server.xml* file located under *drive:\ibi\tomcat\conf*.
2. Add *useBodyEncodingForURI="true"* to the Connector as shown in the following image.

```

67      | Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
68      | -->
69      | <Connector port="8080" protocol="HTTP/1.1"
70      |         connectionTimeout="20000"
71      |         redirectPort="8443"
72      |         maxPostSize="-1"
73      |         useBodyEncodingForURI="true" />
74      | <!-- A "Connector" using the shared thread pool-->

```

Note: WebFOCUS Release 8.1 Version 05 uses Tomcat version 8. In Tomcat version 8, the default URI encoding is UTF-8. If you installed WebFOCUS Release 8.1 Version 05 with Tomcat version 8 and you want to use a Latin 1 client configuration, you must change the URI encoding in Tomcat to Latin 1. For Windows installations, this should be *Cp1252*, as shown in the following example:

```

<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
URIEncoding="Cp1252"
maxPostSize="-1" />

```

For UNIX, this value should be *ISO-8859-1*.

3. Recycle the Tomcat Application Server to activate.

Unicode PDF Output

Reporting from a Unicode data source with PDF output format is available when using the following two fonts which support Unicode characters:

- ❑ Lucida Sans Unicode, which is used to display single-byte characters only.
- ❑ Arial Unicode MS, which is used to display both single-byte and double-byte characters.

The Lucida Sans Unicode font is the default font if the WebFOCUS Server is configured for UTF-8 (code page 65001) or UTF-EBCDIC (code page 65002).

If the server is configured for Unicode, and you want to use the Arial Unicode MS font instead, you must specify the Arial Unicode MS font in the StyleSheet. Alternatively, if the WebFOCUS PDF font mapping file (*drive:\ibi\srn\wfs\etc\fontuser.xml*, where *nn* is your version of the Reporting Server) has *default="yes"* specified for Arial Unicode MS, then it becomes the default font. For example:

```
<family name="Arial Unicode MS">
  <font style="normal"
    metricsfile="pdarum"    default="yes" />
  <font style="bold"
    metricsfile="pdarumb"  />
  <font style="italic"
    metricsfile="pdarumi"  />
  <font style="bold+italic"
    metricsfile="pdarumbi" />
</family>
```

For more information about how to use the PDF font mapping file, see the *Creating Reports With WebFOCUS Language* manual.

Troubleshooting

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

WebFOCUS provides many features that help you perform problem analysis. For example, you can use the WebFOCUS Reporting Server Console and WebFOCUS Administration Console to view current code page settings. In addition, the Transcoding Services Generation Utility (TSGU) enables you to view many NLS values for the current configuration.

Information Builders provides technical support for its world-wide customer base. See Customer Support in the Preface of this manual for information, or contact your local Information Builders representative for details on the full range of support services.

In this chapter:

- ☐ [Troubleshooting Strategy](#)
 - ☐ [Identifying NLS Issues](#)
 - ☐ [Determining NLS Configuration Values](#)
 - ☐ [Examining the Code Page Generation File](#)
 - ☐ [Changing NLS Configuration Settings](#)
-

Troubleshooting Strategy

Information Builders products are designed to work with a variety of computer hardware, third-party database management systems, and data communications networks. A typical WebFOCUS installation may use multiple computers running on different operating systems, and may access and integrate data from a variety of proprietary data sources. Additionally, it may use more than one data communications infrastructure to link the data sources and the server and client computers.

Installations can be complex, and an effective troubleshooting strategy can assist you in isolating a problem to a specific area of the product.

Questions to Ask

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

- ☐ Did you encounter any error messages?

Error messages typically identify 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) generally indicate a problem on a reporting server and are followed by a short description of the problem. You can see the full text of an error message by referring to an errors file or by displaying the message in your session. For instructions, see the *Error Messages* appendix in the *Describing Data With WebFOCUS Language* manual.

It is important to record all error messages, as the information they contain is helpful when you obtain 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 have an important clue to understanding the problem. The next step is to investigate what has changed with the configuration.

- ☐ Has something changed with the system configuration?

By determining what has changed with the configuration, you may be able to solve, or at least isolate, the problem. For example, was something added to 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 EMPLOYEE, or with a sequential file?

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

Find out if a new version of the product was installed since the product last functioned properly, or if the product was 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 browser, a different version of a third-party data source, or a change in the data communications infrastructure could be relevant to the problem.

Observing Results

If the product worked in the past but 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 to the initial settings. If that is possible, change one item at a time and then observe the result.

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 problem and cause.

Additional Resources

The problems you encounter may have already been documented. Information Builders products, such as WebFOCUS, have Release Notes, which are a compilation of all known problems and limitations for a specific release of the software. Additionally, the technical content for the product 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 the most current information on known problems.

Finally, both Information Builders and most third-party vendors have websites with useful support information.

For more information on Information Builders resources available to help you troubleshoot, see [Information You Should Have](#) on page 12.

Identifying NLS Issues

An NLS problem usually manifests itself in one way: an expected national character appears incorrectly (for example, as another national character, an outline box, or an unexpected symbol).

Additionally, an NLS problem is typically specific. For example, text may be garbled in only one feature of a product, or on a particular line in a particular window. It may require a certain sequence of steps to reliably reproduce the problem.

When identifying an NLS problem, keep in mind the following:

- ☐ Application components must be stored with a consistent encoding scheme (that is, with consistent ANSI code pages). For example, the format of a procedure and Master File must be the same as the format of the data sources on the WebFOCUS Reporting Server. In WebFOCUS architecture, it is possible to have two different encoding schemes: a client may have one encoding scheme and pass a procedure to a WebFOCUS Reporting Server for execution with a different encoding scheme.

If the client and server are using different encoding schemes, national characters may not appear correctly or you may receive erroneous output from a report request.

- ☐ The problem may originate with any of the products that work together to support a WebFOCUS installation. Along with Information Builders products, the following may also exhibit NLS problems under certain circumstances:

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

What Should You Do?

When you encounter an NLS problem, 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 the computers, versions of all software (including browser version), and code page settings.

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

Determining NLS Configuration Values

When a WebFOCUS installation is properly configured for NLS, it handles the supported local languages for all features and situations. However, configuration errors can cause NLS problems.

A typical NLS problem is the display of national characters on your PC, or in print, as something other than what you expect. For example, you may think you have properly configured the server and client 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 server (data access) and client code pages. You can verify the code pages using the following:

- ☐ WebFOCUS Reporting Server Console and WebFOCUS Administration Console
- ☐ edaprint.log file (for server verification only)
- ☐ TSGU INFO option of the Transcoding Services Generation Utility (TSGU), supplied with your WebFOCUS software, or the ? LANG command, which shows the same information
- ☐ Server trace file

If you have a complex WebFOCUS Reporting Server with subservers configuration (multiple servers in a nested configuration), you need to determine the code page setting of each server, and verify that the corresponding client code page settings for each server and subservers interface match up correctly.

For examples of server and subservers configurations, see [Information Builders NLS API](#) on page 21.

Procedure: How to Display Server and Client Code Pages From the Consoles (All Platforms)

You can display the current code page for the WebFOCUS Reporting Server and WebFOCUS Client from the applicable console. You must be a server administrator to use the WebFOCUS Reporting Server Web Console and an administrator to use the WebFOCUS Administration Console.

For instructions on accessing the WebFOCUS Reporting Server Web Console, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59.

To display the server code page:

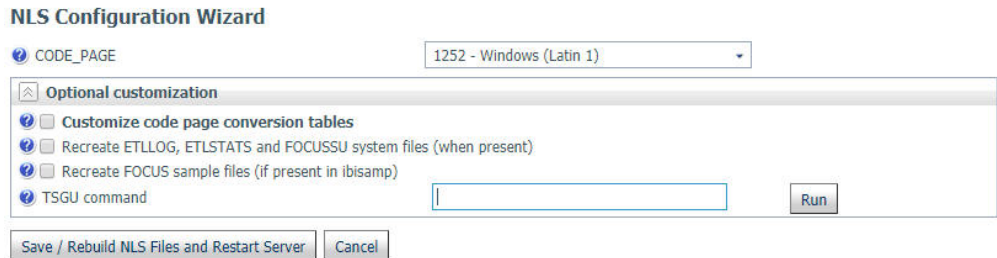
1. Sign in to WebFOCUS Reporting Server Web Console. You must be a server administrator to sign in.

2. Click *Workspace* on the menu bar.

The *Workspace* tab opens.

3. On the ribbon, click *LOCALE* and then click *Configuration Wizard*.

The NLS Configuration Wizard pane opens, as shown in the following image.



Procedure: How to Display the Client Code Page

You must be an administrator to use the WebFOCUS Administration Console.

1. Sign in to the WebFOCUS Administration Console.

- Click *NLS Settings* in the Configuration pane. The code page is selected in the list box on the page, as shown in the following image.

Select the operating system where the WebFOCUS Client resides

- ☐ User Defined Code Page
☒ Windows, UNIX and AS/400
☐ OS/390

Select the WebFOCUS Client Code Page from the list

137 - U.S. English/Western European
 874 - Thai
 942 - Japanese
 946 - Simplified Chinese
 949 - Korean
 1250 - Eastern European
 1251 - Russian
 1252 - Western European
 1253 - Greek
 1254 - Turkish
 1255 - Hebrew
 1256 - Arabic
 1257 - Baltic
 10942 - Japanese EUC
 10948 - Traditional Chinese
 65001 - Unicode (UTF-8)

Save

Cancel

- If necessary, modify the settings to match your operating system and desired code page.
- Click Save.

The highlighted code page is active.

Procedure: How to Run TSGU INFO or ? LANG From the Console (All Platforms)

The TSGU INFO and ? LANG commands return the value of many NLS settings, including server and client code page, Continental Decimal Notation, currency symbol, and others. You must be a server administrator to use the WebFOCUS Reporting Server Console.

For instructions on accessing the WebFOCUS Reporting Server Console, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59.

1. Sign in to the WebFOCUS Reporting Server Console.
2. Click *Applications* on the menu bar.
3. Right-click an application folder, point to *New*, and click *Procedure*.
4. To retrieve information on the NLS configuration at your site, type

`TSGU INFO`

or

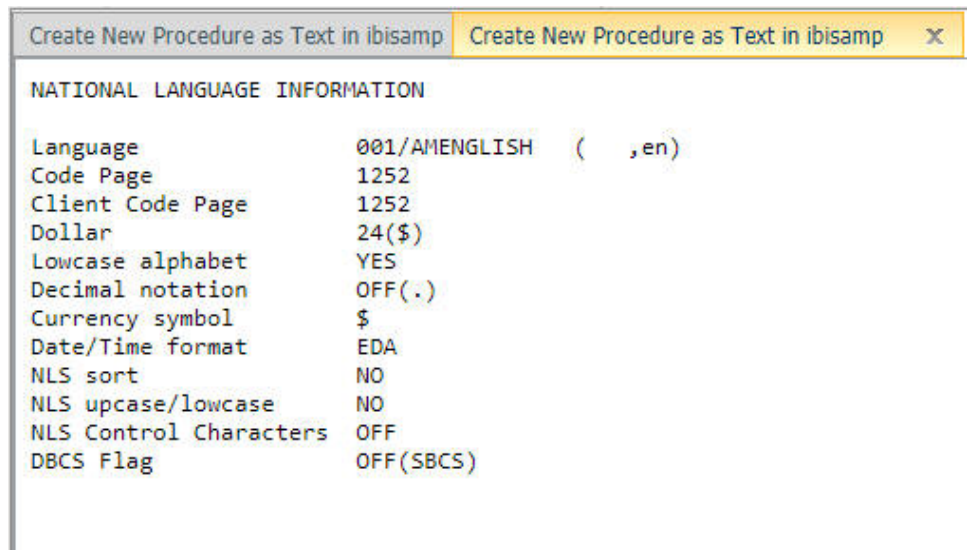
`? LANG`

into the text editor.

5. Click the *Run* button.

The results open in a new tab in the Web Console.

The following is sample information from the TSGU INFO command. The same National Language Information report appears if you run the `? LANG` command. For a description of the values returned, see [Interpreting NLS Information](#) on page 113.



The screenshot shows a web browser window with two tabs. The active tab is titled "Create New Procedure as Text in ibisamp". The content of the tab is a text editor displaying the output of the "TSGU INFO" command. The output is titled "NATIONAL LANGUAGE INFORMATION" and lists various NLS settings and their values.

NATIONAL LANGUAGE INFORMATION		
Language	001/AMENGLISH	(,en)
Code Page	1252	
Client Code Page	1252	
Dollar	24(\$)	
Lowcase alphabet	YES	
Decimal notation	OFF(.)	
Currency symbol	\$	
Date/Time format	EDA	
NLS sort	NO	
NLS upcase/lowcase	NO	
NLS Control Characters	OFF	
DBCS Flag	OFF(SBCS)	

6. Close the tab to return to the text editor, and optionally, click *Save As* to save the procedure for reuse.

Reference: Interpreting NLS Information

Setting	Description
Language	Is the language code (for example, 001) and the name of the language for server error messages (for example, AMENGLISH) on the WebFOCUS Reporting Server.
Code Page	Is the code page configured on the WebFOCUS Reporting Server for the language of the data sources.
Client Code Page	Is the code page configured on the WebFOCUS Client for the correct display of report output in the browser.
Dollar	Is the code point (for example, 24) and symbol (for example, \$) for the dollar sign when it is used as a delimiter in certain WebFOCUS files, such as Master Files.
Lowcase alphabet	Indicates if lowercase alphanumeric characters are supported.
Decimal notation	Is the setting for Continental Decimal Notation. For possible values, see Punctuating Large Numbers on page 129.
Currency symbol	Is the character that identifies the currency symbol. For possible values, see Extended Currency Symbol Formats on page 133.
Date/Time format	Is the date/time format for Structured Query Language (SQL). For possible values, see NLS Configuration File Parameters on page 66.
NLS sort	Indicates if WebFOCUS created the file that contains the sorting tables (sorttbl.err on Windows, UNIX, and IBM i). If you configured the WebFOCUS Reporting Server for NLS from the console, or if you manually ran the TSGU SORT command, the value is YES. Otherwise, the value is NO.
NLS upcase/lowcase	Indicates if WebFOCUS created the file that contains the monocasing tables (casetbl.err on Windows, UNIX, and IBM i). If you configured the WebFOCUS Reporting Server for NLS from the console, or if you manually ran the TSGU CASE command, the value is YES. Otherwise, the value is NO.

Setting	Description
NLS Control Characters	Reserved for future use.
DBCS Flag	<p>Indicates if a Double-Byte Character Set (16-bit encoding) is used, and if so, which alphabet is enabled. If a DBCS is used, the value is ON, followed by the identifier for a Japanese or Taiwanese alphabet, for example, ON(SJIS). For possible values, see the comment for DBCS - DBCS ID in Supported Code Page File on page 68.</p> <p>If a DBCS is not used, the value is OFF(SBCS).</p>

Procedure: How to Start and Display Server Trace From the Console (All Platforms)

You can determine the server (data access) and client code pages by examining the server trace file. You must be a server administrator to use the WebFOCUS Reporting Server Console.

For instructions on accessing the WebFOCUS Reporting Server Console, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59.

1. Sign in to the WebFOCUS Reporting Server.
2. Click *Workspace* in the menu bar. In the navigation pane, expand *Logs and Traces* and open *Traces*.
3. If you have never enabled tracing before, the Traces window indicates that no traces have been found.

In the navigation pane, right-click *Traces* and click *Enable Traces* to turn tracing on.

Tip: The trace file can be very large. Selecting filters makes the trace file easier to navigate. To filter trace file contents:

- a. To filter the traces, right-click *Traces* and click *Configure*.

The Traces Configuration screen opens, as shown in the following image.

Traces Configuration

Setting will be activated for next workspace run

Set number of lines to limit the trace size

(leave blank for no limit)

Choose the trace components to activate:

Save

Save and Restart Server

- b. From the Choose the trace components to activate menu, select *Custom Components*. A list of components appears.
- c. Use the checkbox at the top of the leftmost column to deselect all components.
- d. Scroll down to NLS Services and select the two NLS options.

<input type="checkbox"/>	▼ NLS Services	
<input type="checkbox"/>	GX	NLS /1 - NLS API Call Trace
<input type="checkbox"/>	GY	NLS /2 - NLS Data Conversion

- e. Click Save.

Traces Configuration

Setting will be activated for next workspace run

Set trace size limit to lines (leave blank for no limit)

Choose the trace components to activate:

☐ Default Components
☐ All Components
☐ Typical Components
☒ Custom Components

4. On the ribbon, in the Diagnostic group, click *Log and Trace* and click *Last Agent Trace*.

The trace file is downloaded in the browser.

5. Open the file using a text editor, such as Notepad.
6. The trace file displays in a new window. To display the client and server code page, search for the text string:

```
nwnlscon: client code page
```

Reference: Sample Server Trace File

This is part of a sample server trace file (ts000011.trc). The first line indicates that the client code page is 1252 and the server (agent) code page is 1252.

The trace file contains additional valuable information. To assist you in troubleshooting, an Information Builders Customer Support Consultant may ask you to provide the full contents of this file, an example of which is shown in the following image.

```
15.45.48 AI nwnlscon:          client code page = 1252 ---> agent code page = 1252
15.45.48 GX nlsql:    NLS TRACE is activated in nlsql
15.45.48 GX nlsql:    intlcm=000001F555B5A6E0  codepg=1252
15.45.48 GY nlsql:    Client Code Page: 1252 sqlval=24($)
```

Examining the Code Page Generation File

The WebFOCUS Reporting Server maintains a code page generation file. When you select a code page for the server using the WebFOCUS Reporting Server Console, WebFOCUS updates this file with the selected code page and with the associated client code pages.

The NLS API uses the code pages listed in this file to generate the transcoding, monocasing, and sorting tables for the WebFOCUS Reporting Server and WebFOCUS Client.

On Windows, UNIX, and IBM i, the code page generation file, `cpcodepg.nls`, resides in the `nls` directory under the home directory (for example, on Windows, `drive:\ibi\svr77\home\nls`). On z/OS, the file is member `CPCODEPG` in the data set `qualif.EDACTL.DATA`.

If you experience a problem associated with NLS, make sure that the server code page and client code page are listed in this file. If they are not, use the WebFOCUS Reporting Server Console and WebFOCUS Administration Console to configure the server and client. For instructions, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59 and [Configuring a WebFOCUS Client for NLS](#) on page 79.

Changing NLS Configuration Settings

Once you determine that an error occurred in the NLS configuration, and that the server or client code page is not what you intended, you can correct the configuration by changing the code page settings.

For instructions on changing the code page settings, see [Configuring the WebFOCUS Reporting Server for NLS](#) on page 59 and [Configuring a WebFOCUS Client for NLS](#) on page 79.

Display, Print, and Language Considerations

This topic describes display, print, and WebFOCUS Language considerations for NLS.

In this chapter:

- ☐ [Display Issues](#)
 - ☐ [Print Issues](#)
 - ☐ [WebFOCUS Language Considerations](#)
-

Display Issues

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

- ☐ Browsers typically support 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. If this is the case, you must reset the browser for each encoding option for the target local language.
- ☐ If you cannot read some Unicode characters in the browser, your system may not be properly configured. Confirm that you are using the most recent version of browser software and that you installed the necessary fonts. For information on installing browser fonts, refer to the browser documentation.

- ❑ WebFOCUS supports PDF and PS output for the following languages: Baltic, Traditional and Simplified Chinese, Japanese, Hebrew, Thai, Western (English, French, Italian, German, Spanish, Portuguese) and Central European (Turkish, Bulgarian, Czech, Estonian, Greek, Hungarian, Latvian, Lithuanian, Polish, Russian, and Ukrainian). However, due to copyright laws, only Western European language fonts are shipped with the product. Customers must provide their own PostScript Type 1 Fonts for PS and PDF Formats to display NLS characters in other supported languages. For information on how to add/configure these fonts in a WebFOCUS StyleSheet see the *Choosing a Display Format* chapter in the *Creating Reports With WebFOCUS Language* manual.
- ❑ Default fonts used for PDF and PS output formats for Hebrew and Arabic are based on the server language (LANG) parameter. Assuming that the StyleSheet does not specify a font name and that no default font is set in the PDF font mapping file, the defaults are Hebrew (Lucinda Sans Unicode) and Arabic (Traditional Arabic).

Print Issues

The earliest computer printers used mechanisms like chains, daisy wheels, or type balls, which printed (impressed) individual characters on paper output. For these technologies, the print mechanism must include the national characters you want to print.

For example, if you are using a traditional mainframe line printer with a print chain to produce reports, and you want to print the euro symbol (€) for the European Community common currency, the print chain will have to include that symbol.

However, today's computer printers typically use technologies such as ink jet or laser printing that are able to print any shape, not just text. Available characters are determined by printer firmware (fixed software routines embedded in specialized microchips), or software controlling the printer.

Note the following issues involving printing technologies and NLS:

- ❑ To correctly print a script normally not used by a printer, you may need to specify bitmap images for text or something similar.
- ❑ 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 for printing is parsed symbol by symbol, and the nearest matching symbol in a system font is used. This system works for most applications, and improves performance. However, system fonts may not support the national characters you need.

In these cases and in others you may encounter, consult the printer documentation for information on how to proceed. Additionally, consult the documentation for your operating system and network. Print options are often under operating system or network control.

WebFOCUS Language Considerations

If your application does not behave as expected, make sure your code complies with the following WebFOCUS language conventions:

❑ Double-Byte Character Sets

Limitations apply to certain WebFOCUS features that involve character input. For example, in a single report, you can enter a maximum of 32K characters for headings and footings.

If you are using a Double-Byte Character Set (DBCS), cut the limitation in half. Using the same example, a maximum of 16K characters from a DBCS is allowed for headings and footings in one report.

❑ Field Names and Aliases

Do not use the following characters in a field name or alias on a WebFOCUS Reporting Server running on z/OS (USS) or IBM i. WebFOCUS interprets each of these characters as the terminating dollar sign (hexadecimal 5B) used in Master Files, Access Files, and WebFOCUS StyleSheet files, causing unexpected results.

Server Code Page	Character	Country
277	Å (capital A with ring above)	Denmark, Norway
278	Å (capital A with ring above)	Finland, Sweden
285	£ (pound sign)	United Kingdom
1026	İ (capital I with dot above)	Turkey

History of Code Pages

This appendix discusses the evolution of code pages and their standardization by a number of organizations. It describes the double-byte encoding schemes, originating in Japan, that have been developed for Far Eastern languages.

It also describes the implementation of Unicode (a double-byte encoding scheme) as a code page architecture, and includes a summary of code page families used by Information Builders products.

In this appendix:

- ☐ [Origin of Code Pages](#)
 - ☐ [Far Eastern Encodings](#)
 - ☐ [Unicode](#)
 - ☐ [Code Page Families](#)
-

Origin of Code Pages

The computer code page architectures of today are the direct descendants of earlier text processing technologies. Specifically, binary and numeric codes for individual letters have been used since the earliest days of the telegraph and teletype. Punched card technology, the immediate predecessor of modern electronic computers, also used binary and numeric codes.

In the 1950s, 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 values) 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, in which 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 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 in 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 processing 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 ASCII implementation among the vendors.

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

Far Eastern Encodings

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

The general encoding solution for the CJK scripts was developed in Japan. 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 schemes are used for CJK encodings.

The first scheme, modal encoding, uses a two-stage process. The first stage is mode switching (for example, switching 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.

The second scheme, non-modal encoding, uses the numeric byte value of a text stream 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-5 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 scheme, fixed-width encoding, uses 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 a large amount of space. UTF-16 Unicode is an example of a fixed-width encoding scheme.

These schemes are generally applied to actual code page implementation by an operating system-specific method. Some systems use both a single-byte code page (for Latin text 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

In the 1990s, the Unicode Consortium (an industry standards group) and ISO 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 and Japanese.

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 Information Builders, now provide Unicode data types. Many computer and web technologies use Unicode for text data, including Java technologies, HTML, the XML family of languages, and Microsoft Windows and Office.

Employing Unicode for text data in computer technologies is a growing trend. Specifically, the operating system components of smart phones and tablets often use Unicode for all text data.

Code Page Families

Today, with 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 summary will help you understand the organization of code pages.

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 used by Information Builders products are:

- ❑ Microsoft Windows code pages.
- ❑ UNIX code pages. These code pages include the ISO 8859-x set, which is similar to Microsoft Windows code pages.
- ❑ IBM EBCDIC code pages. These code pages are used for platforms such as IBM z/OS and IBM i.
- ❑ 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 for legacy applications.

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 Windows and UNIX 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 and UNIX. Since Release 7.7, however, code page 1252 has been used as the default for ASCII platforms.

Helpful NLS Commands

WebFOCUS provides many commands that are helpful in your implementation of National Language Support (NLS). These commands govern features such as the display of local language error messages, the language and formatting of Excel, and the formatting of numbers.

In addition, a field formatting feature allows you to selectively display certain currency symbols in individual reports regardless of NLS configuration. Other features enable you to specify report titles and metadata in different languages in a Master File.

For more information on the SET commands described in this section, see the *Customizing Your Environment* chapter in the *Developing Reporting Applications* manual.

This appendix also contains procedures for the generation and display of national characters.

The commands in this section apply to all operating systems unless otherwise noted.

In this appendix:

- ☐ [Setting the Language of Excel in the NLS Configuration File](#)
 - ☐ [Setting the Excel Version Number](#)
 - ☐ [Setting Column Width for Excel in the NLS Configuration File](#)
 - ☐ [Punctuating Large Numbers](#)
 - ☐ [Displaying a Leading Zero in Decimal-only Numbers](#)
 - ☐ [Selecting an Extended Currency Symbol](#)
 - ☐ [Setting Date and Time Display and Formatting](#)
 - ☐ [Setting Business Days and Holidays](#)
 - ☐ [Determining Collation of Data](#)
 - ☐ [Displaying National Characters in Server-Side Graphics on UNIX and Linux](#)
 - ☐ [Specifying Multilingual Metadata in a Master File](#)
 - ☐ [Storing Localized Metadata in Language Files](#)
 - ☐ [Adding the Code Page Parameter to a Master File](#)
 - ☐ [Running PDF or PS Reports with International Fonts in App Studio](#)
-

Setting the Language of Excel in the NLS Configuration File

The WebFOCUS Reporting Server needs to know the language of Excel running on the computer of the end user in order to correctly format output returned to Excel. You specify the language on the EXL2KLANG parameter in the NLS configuration file (nlscfg.err).

You can also code the SET EXL2KLANG command in a server profile or user profile, or in a procedure, to override the setting in the NLS configuration file.

Syntax: How to Set the Language of Excel in the NLS Configuration File

```
EXL2KLANG = { language | ENG }
```

where:

language

Is the language of Excel. Possible values are:

- ☐ **ENG** for English. ENG is the default value.
- ☐ **FIN** for Finnish.
- ☐ **FRE** for French.
- ☐ **GER** for German.
- ☐ **JPN** for Japanese.
- ☐ **NOR** for Norwegian.
- ☐ **POL** for Polish.
- ☐ **PRC** for Simplified Chinese.
- ☐ **SPA** for Spanish.
- ☐ **ROC** for Traditional Chinese.

Example: Controlling Numeric Format in Excel Through Regional Options

In this example, a WebFOCUS report request accesses a data source that contains numbers stored in English (United States) format. The user has set Regional Options on Windows to the locale German (Germany).

The report output displays in an Excel spreadsheet, as shown in the following image. The numbers are formatted according to the convention of the German locale:

	A	B	C	D	E
1	Short Term Investments				
2	Excel 2000 Spreadsheet				
3					
4	Fund Manager	Type of Instrument	Instrument Holder	Fund Balance	
5	01005	CASH	COMM	\$7.200.830,00	
6	01005	CASH	GOVT	\$144.575.691,00	
7	01005	OVERNIGHT	GOVT	\$79.670.291,00	
8	01005	ST. NOTES	GOVT	\$221.001.921,00	
9	02005	OVERNIGHT	COMM	\$181.100.192,00	
10	02005	ST. NOTES	COMM	\$441.772.390,00	
11					
12					
13					
14					
15					
16					

Setting the Excel Version Number

When using the EXL2K PIVOT format with international versions of Excel, you can set the EXCELRELEASE parameter in the edasprof.prf or nlscfg.err files, or in the procedure itself, to distinguish the version of the user Excel application. If you do not know which version of Excel the users have, you can add EXCELRELEASE as an amper variable within the procedure so that the users are prompted to select their version of Excel. The EXL2KLANG parameter must also be set to the same language as the user version of Excel.

Syntax: How to Set the Excel Version in the NLS Configuration File

```
EXCELRELEASE = release
```

where:

release

Is the version of the user Excel application. Values can be: 2000, 2002, or 2003.

Syntax: How to Set the Excel Version in a Report Request

To set the version of Excel in a report request, use the following syntax:

```
TABLE FILE data_source_name
. . .
ON TABLE SET EXCELRELEASE release
. . .
END
```

where:

data_source_name

Is the name of the data source being reported against.

release

Is the version of the user Excel application. Values can be: 2000, 2002, or 2003.

Example: Setting the Excel Version in a Report Request

The following report request against the CAR database sets the Excel version to Excel 2003:

```
TABLE FILE CAR
PRINT DEALER_COST
BY CAR
ON TABLE SET EXCELRELEASE 2003
ON TABLE PCHOLD FORMAT EXL2K PIVOT
END
```

Setting Column Width for Excel in the NLS Configuration File

The EXL2K_DEFCOLW parameter in the NLS configuration file tells WebFOCUS the value to use to format the output returned to Excel. It is a fixed value for each language.

WebFOCUS automatically uses the correct value for certain languages, and you do not need to do anything:

- ☐ For English, Finnish, Japanese, Polish, Simplified Chinese, and Traditional Chinese, WebFOCUS uses the correct value of 8 (the default).
- ☐ For French, German, Norwegian, and Spanish, WebFOCUS uses the correct value of 10.

If you are using a language for Excel other than one mentioned here and you experience a problem with Excel pivot tables or column width, contact your Information Builders representative.

Syntax: How to Set Column Width for Excel in the NLS Configuration File

```
EXL2K_DEFCOLW = nn|8
```


where:

nn

Is the column width for Excel. 8 is the default value.

Punctuating Large Numbers

Countries differ in the way they punctuate numbers. You can reflect these differences in WebFOCUS reports using Continental Decimal Notation (CDN), specified with the SET CDN command. This command enables you to control the punctuation of large numbers, using commas, decimals, spaces, or single quotation marks.

You can use the SET CDN command in a report request but not in a DEFINE or COMPUTE command.

The punctuation specified by the SET CDN command also determines the punctuation used in numbers affected by the SET CENT-ZERO command. For more information on that command, see [Displaying a Leading Zero in Decimal-only Numbers](#) on page 131.

For Excel 2000 or later: If the display format of a WebFOCUS report is Excel 2000 or later, Continental Decimal Notation is controlled by the settings on an end user computer. That is, numbers in report output are formatted according to the convention of the locale (location) set in regional or browser language options.

Syntax: How to Punctuate Large Numbers

SET CDN = option

where:

option

Is one of the following:

- ☐ **ON** or **DOTS_COMMA**, which uses CDN. ON sets the decimal separator as a comma and the thousands separator as a period. For example, the number 3,045,000.76 is represented as 3.045.000,76. ON should be used for Germany, Denmark, Italy, Spain, and Brazil.
- ☐ **OFF** or **COMMAS_DOT**, which turns CDN off. For example, the number 3,045,000.76 is represented as 3,045,000.76. OFF is the default value. OFF should be used for the USA, Canada, Mexico, and the United Kingdom.

- ❑ **SPACE** or **SPACES_COMMA**, which sets the decimal point as a comma, and the thousands separator as a space. For example, the number 3,045,000.76 is represented as 3 045 000,76. SPACE should be used for France, Norway, Sweden, and Finland.
- ❑ **QUOTE** or **QUOTES_COMMA**, which sets the decimal point as a comma and the thousands separator as an apostrophe. For example, the number 3,045,000.76 is represented as 3'045'000,76. QUOTE should be used for Switzerland.
- ❑ **QUOTEP** or **QUOTES_DOT**, which sets the decimal point as a period and the thousands separator as an apostrophe. For example, the number 3,045,000.76 is represented as 3'045'000.76.

Note: If the display format of a report is Excel 2000 or later, Continental Decimal Notation is controlled by the settings on the user computer. That is, numbers in report output are formatted according to the convention of the locale (location) set in regional or browser language options.

Example: Punctuating Large Numbers

In the following request, CDN is set to ON, which punctuates numbers using a period to separate thousands, and a comma to separate the decimal value.

```
SET CDN = ON
SET PAGE-NUM = OFF
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME SALARY
ON TABLE SET STYLE *
TYPE = REPORT, GRID = OFF,$
ENDSTYLE
END
```

The partial output is:

<u>LAST NAME</u>	<u>FIRST NAME</u>	<u>SALARY</u>
STEVENS	ALFRED	\$11.000,00
STEVENS	ALFRED	\$10.000,00
SMITH	MARY	\$13.200,00
JONES	DIANE	\$18.480,00
JONES	DIANE	\$17.750,00
SMITH	RICHARD	\$9.500,00
SMITH	RICHARD	\$9.050,00
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

Displaying a Leading Zero in Decimal-only Numbers

By default, when a number containing only a decimal portion appears in a report, a leading zero is not displayed. Use the SET CENT-ZERO command to display a leading zero in decimal-only numbers.

The setting of CDN determines the characters used in the punctuation of a number. For more information on CDN, see [Punctuating Large Numbers](#) on page 129.

Syntax: How to Display a Leading Zero in Decimal-only Numbers

```
SET CENT-ZERO = {ON|OFF}
```

where:

[ON](#)

Displays a leading zero in decimal-only numbers.

[OFF](#)

Suppresses a leading zero in decimal-only numbers. OFF is the default value.

Example: Displaying a Leading Zero in Decimal-only Numbers

In the following request, CENT-ZERO is set to ON, which displays a leading zero in decimal-only numbers:

```
SET CENT-ZERO = ON
SET PAGE-NUM = OFF
TABLE FILE CENTINV
PRINT PRODNAME
COMPUTE FACTOR = COST/RETAIL;
BY PRODCAT
WHERE PRODCAT EQ 'Cameras';
ON TABLE SET STYLE *
TYPE = REPORT, GRID = OFF,$
ENDSTYLE
END
```

The output is:

Product		FACTOR
Product Category:	Name:	
Cameras	340SX Digital Camera 65K P	0.80
	330DX Digital Camera 1024K P	0.71
	AR3 35MM Camera 10 X	0.74
	AR2 35MM Camera 8 X	0.72

Selecting an Extended Currency Symbol

You can select a currency symbol for display in report output regardless of the default currency symbol configured for National Language Support (NLS). Use the extended currency symbol format in place of the floating dollar (M) or non-floating dollar (N) display option in a Master File or in a DEFINE command in a procedure.

When you use the floating dollar (M) or non-floating dollar (N) display option, WebFOCUS prints the currency symbol associated with the default code page. For example, when you use an American English code page, the dollar sign is displayed as the currency symbol.

The extended currency symbol format allows you to display a symbol other than the dollar sign. Using the extended currency symbol format, you can display the symbol for a United States dollar, a British pound, a Japanese yen, or the euro.

The extended currency symbol formats are available for numeric formats (I, D, F, and P).

Reference: Extended Currency Symbol Formats

The following guidelines apply:

- ☐ A format specification cannot be longer than eight characters.
- ☐ The extended currency option must be the last option in the format.
- ☐ The extended currency symbol format cannot include the floating (M) or non-floating (N) display option.
- ☐ A non-floating currency symbol is displayed only on the first row of a report page. If you use field-based reformatting (as in the example that follows) to display multiple currency symbols in a report column, only the symbol associated with the first row is displayed. In this case, do not use non-floating currency symbols.

Syntax: How to Use an Extended Currency Symbol Field Format

You can also set an extended currency symbol by specifying it in the format of a field. You can set a currency symbol field format in a DEFINE or COMPUTE field or in a function.

numeric_format{:|!}option

where:

numeric_format

Is a valid numeric format (data type I, D, F, or P).

{:|!}

Either a colon (:) or exclamation point (!) is required. Note that only the colon is invariant across code pages, and therefore may be preferable.

option

Determines the currency symbol that is displayed and where the symbol appears. Possible values are:

- ☐ **d** displays a non-floating dollar sign.
- ☐ **D** displays a floating dollar sign.
- ☐ **e** displays a non-floating euro symbol.
- ☐ **E** displays a floating euro symbol.
- ☐ **F** displays a euro symbol to the right of the number.

- ☐ **G** displays a dollar sign to the right of the number.
- ☐ **I** displays a non-floating British pound sterling symbol.
- ☐ **L** displays a floating British pound sterling symbol.
- ☐ **y** displays a non-floating Japanese yen symbol.
- ☐ **Y** displays a floating Japanese yen symbol.

Example: Displaying Extended Currency Symbols

The following request displays the British pound sterling symbol on the row that represents England, the euro symbol on the row that represents Italy, and the Japanese yen symbol on the row that represents Japan.

```
SET PAGE-NUM = OFF
DEFINE FILE CAR
CFORMAT/A8 = DECODE COUNTRY('ENGLAND' 'F12.1C!L' 'JAPAN' 'D12!Y'
                           ELSE 'D12.2!E');
END

TABLE FILE CAR
PRINT SALES/CFORMAT DEALER_COST/CFORMAT
BY COUNTRY
  WHERE COUNTRY EQ 'ENGLAND' OR 'JAPAN' OR 'ITALY'
  WHERE SALES GT 0
ON TABLE SET STYLE *
TYPE = REPORT, GRID = OFF,$
ENDSTYLE
END
```

The output is:

<u>COUNTRY</u>	<u>SALES</u>	<u>DEALER COST</u>
ENGLAND	£12,000.0	£11,194.0
ITALY	€4,800.00	€4,915.00
	€12,400.00	€5,660.00
	€13,000.00	€5,660.00
JAPAN	¥43,000	¥2,626
	¥35,030	¥2,886

You can also use an extended currency symbol in a report by applying it from the user interface in InfoAssist+ or App Studio.

☐ **In InfoAssist+:**

- ☐ Select a numeric field in your report. On the Field tab, in the Format group, open the Change currency options drop-down menu. You can select a currency symbol and where to position it.
- ☐ Select a numeric field in your report. On the Field tab, in the Format group, open the main drop-down menu and click *More options....* The Field Format Options dialog box opens. Open the Currency Symbol menu and select an option. You can select a currency symbol and where to position it, as well as set other formatting options.

☐ **In App Studio:**

- ☐ Select a numeric field in your report. On the Appearance tab, in the Format group, open the Currency drop-down menu. You can select a currency symbol and where to position it.

Reference: Specifying an Extended Currency Symbol

The CURRSYMB parameter specifies a symbol used to represent currency when a numeric format specification uses the M or N display options. The default currency symbol depends on the code page being used.

The syntax is:

```
SET CURRSYMB = symbol
```

where:

symbol

Is any printable character or a supported currency code. The following are possible values.

Note: In order to specify a dollar sign (\$) as the character, you must enclose it in single quotation marks (').

- ☐ **USD or '\$'.** Specifies U.S. dollars.
- ☐ **GBP.** Specifies the British pound.
- ☐ **JPY.** Specifies the Japanese yen.
- ☐ **EUR.** Specifies the euro.

- ❑ **NIS.** Specifies the Israeli new sheqel.

Reference: Setting the Symbol for D or d Formats

The CURSYM_D parameter specifies the characters used to represent currency when a numeric format specification uses the !D, :D, !d, or :d display options, which by default display a floating (D) or fixed (d) dollar sign to the left of the number.

The syntax is:

```
SET CURSYM_D = symbol
```

where:

symbol

Is up to four printable characters.

Reference: Setting the Symbol for E or e Formats

The CURSYM_E parameter specifies the characters used to represent currency when a numeric format specification uses the !E, :E, !e, or :e display options, which by default display a floating (E) or fixed (e) euro sign to the left of the number.

The syntax is:

```
SET CURSYM_E = symbol
```

where:

symbol

Is up to four printable characters.

Reference: Setting the Symbol for the F Format

The CURSYM_F parameter specifies the characters used to represent currency when a numeric format specification uses the !F or :F display option, which by default displays a floating euro sign to the right of the number. This command supports adding a blank space between the number and the currency symbol.

The syntax is:

```
SET CURSYM_F = symbol
```


where:

symbol

Is up to four printable characters. If the characters include a blank space, they must be enclosed in single quotation marks.

Reference: Setting the Symbol for the G Format

The CURSYM_G parameter specifies the characters used to represent currency when a numeric format specification uses the !G or :G display option, which by default displays a floating dollar sign to the right of the number. This command supports adding a blank space between the number and the currency symbol.

The syntax is:

```
SET CURSYM_G = symbol
```

where:

symbol

Is up to four printable characters. If the characters include a blank space, they must be enclosed in single quotation marks.

Reference: Setting the Symbol for L or I Formats

The CURSYM_L parameter specifies the characters used to represent currency when a numeric format specification uses the !L, :L, !I, or :I display options, which by default display a floating (L) or fixed (I) British pound symbol to the left of the number.

The syntax is:

```
SET CURSYM_L = symbol
```

where:

symbol

Is up to four printable characters.

Reference: Setting the Symbol for Y or y Formats

The CURSYM_Y parameter specifies the characters used to represent currency when a numeric format specification uses the !Y, :Y, !y, or :y display options, which by default display a floating (L) or fixed (I) Japanese yen or Chinese yuan symbol to the left of the number.

The syntax is:

```
SET CURSYM_Y = symbol
```

where:

symbol

Is up to four printable characters.

Using a Currency ISO Code

You can specify an ISO code to display automatically for fields with monetary values. By default, this code only appears as a replacement when the currency symbol cannot be displayed by the code page in effect. However, you can determine whether you want the ISO code to always display or never display instead.

These values can be set for an individual procedure using the syntax below, or can be set at the server level for all procedures run on that server. For more information, see [How to Change LOCALE Settings on the WebFOCUS Reporting Server](#) on page 63.

Reference: Specifying the Currency ISO Code

The CURRENCY_ISO_CODE parameter defines the ISO code for the currency symbol to use. By default, the server uses the currency code that matches the configured language code, but you can set a currency code manually.

The syntax is:

```
SET CURRENCY_ISO_CODE = iso
```

where:

iso

Is a standard three-character currency code, such as USD for US dollars or JPY for Japanese yen. The default value is *default*, which uses the currency code for the configured language code.

Reference: Displaying a Currency ISO Code

You can determine when you want a currency ISO code to display using the CURRENCY_PRINT_ISO parameter. By default, the ISO code only appears when the currency symbol cannot be displayed by the code page in effect.

The syntax is:

```
SET CURRENCY_PRINT_ISO = {DEFAULT|ALWAYS|NEVER}
```

where:

DEFAULT

Replaces the currency symbol with its ISO code when the symbol cannot be displayed by the code page in effect. This is the default value.

ALWAYS

Always replaces the currency symbol with its ISO code.

NEVER

Never replaces the currency symbol with its ISO code. If the currency symbol cannot be displayed by the code3 page in effect, it will not be printed at all.

Note: Using a Unicode environment allows the printing of all currency symbols, otherwise this setting is needed.

Setting Date and Time Display and Formatting

You can use SET commands in a procedure to determine how dates will be displayed and formatted in your content. Some of these parameters can be set in the LOCALE settings on the Web Console to apply automatically to any procedures created on the Reporting Server for which these SET parameters are not defined. For more information, see [How to Change LOCALE Settings on the WebFOCUS Reporting Server](#) on page 63.

DATE_ORDER

This parameter defines the order of date components for display.

The syntax is:

```
SET DATE_ORDER = {DEFAULT | DMY | MDY | YMD}
```

where:

[DEFAULT](#)

Respects the original order of date components. This is the default value.

[DMY](#)

Displays all dates in day/month/year order.

[MDY](#)

Displays all dates in month/day/year order.

[YMD](#)

Displays all dates in year/month/day order.

Note:

- ☐ **DATE_ORDER** and **DATE_SEPARATOR** override the specified date order for all date and date-time displays unless they include a translation display option (T, Tr, t, or tr), in which case the specified order is produced. To limit the scope to a request, use the ON TABLE SET phrase.

- ❑ To use these settings with the Dialogue Manager system variables, (for example, &DATE, &TOD, &YMD, &DATEfmt, and &DATXfmt) append the suffix `.DATE_LOCALE` to the system variable. This allows system variables that are localized to coexist with non-localized system variables.

DATE_SEPARATOR

This parameter defines the separator for date components for display.

The syntax is:

```
SET DATE_SEPARATOR = separator
```

where:

separator

Can be one of the following values.

- ❑ **DEFAULT**, which respects the separator defined by the USAGE format of the field.
- ❑ **SLASH**, which uses a slash (/) to separate date components.
- ❑ **DASH**, which uses a dash (-) to separate date components.
- ❑ **BLANK**, which uses a blank to separate date components.
- ❑ **DOT**, which uses a dot (.) to separate date components.
- ❑ **NONE**, which does not separate date components.

Note:

- ❑ `DATE_ORDER` and `DATE_SEPARATOR` override the specified date order for all date and date-time displays unless they include a translation display option (T, Tr, t, or tr), in which case the specified order is produced. To limit the scope to a request, use the `ON TABLE SET` phrase.
- ❑ To use these settings with the Dialogue Manager system variables, (for example, &DATE, &TOD, &YMD, &DATEfmt, and &DATXfmt) append the suffix `.DATE_LOCALE` to the system variable. This allows system variables that are localized to coexist with non-localized system variables.

TIME_SEPARATOR

This parameter defines the separator for time components for the &TOD system variable.

The syntax is:

```
SET TIME_SEPARATOR = {DOT|COLON}
```

where:

DOT

Uses a dot (.) to separate time components. This is the default value.

COLON

Uses a colon (:) to separate time components.

Setting Business Days and Holidays

You can use SET commands in a procedure to determine what days are considered holidays and business days in your procedures and functions. Some of these parameters can be set in the LOCALE settings on the Web Console to apply automatically to any procedures created on the Reporting Server for which these SET parameters are not defined. For more information, see [How to Change LOCALE Settings on the WebFOCUS Reporting Server](#) on page 63.

BUSDAYS

The BUSDAYS parameter specifies which days are considered business days and which days are not if, your business does not follow the traditional Monday through Friday week.

The syntax is:

```
SET BUSDAYS = {week|_MTWTF_}
```

where:

week

Is SMTWTFS, representing the days of the week. Any day that you do not want to designate as a business day must be replaced with an underscore in the designated place for that day.

If a letter is not in its correct position, or if you replace a letter with a character other than an underscore, you receive an error message. _MTWTF_ is the default value.

HDAY

The HDAY parameter specifies the holiday file from which to retrieve dates that are designated as holidays for use with the date functions DATEDIF, DATEMOV, DATECVT, and DATEADD. The file must be named HDAY, followed by two to four characters.

To clear the holiday file, use

```
SET HDAY = OFF
```

The syntax is:

```
SET HDAY = xxxx
```

where:

```
xxxx
```

Are the letters in the name of the holiday file, named HDAYxxxx. This string must be between two and four characters long.

The default is no setting for this parameter.

WEEKFIRST

The WEEKFIRST parameter specifies a day of the week as the start of the week. This is used in week computations by the HDIFF, HNAME, HPART, HYYWD, and HSETPT functions, described in the *Using Functions* manual.

The HPART and HNAME subroutines can extract a week number from a date-time value. To determine a week number, they can use ISO 8601 standard week numbering, which defines the first week of the year as the first week in January with four or more days. Any preceding days in January belong to week 52 or 53 of the preceding year.

Depending on the value of WEEKFIRST, these functions can also define the first week of the year as the first week in January with seven days.

The WEEKFIRST parameter does not change the day of the month that corresponds to each day of the week, but only specifies which day is considered the start of the week.

The syntax is:

```
SET WEEKFIRST = {value|1}
```

where:

value

Can be:

1 through 7, representing Sunday through Saturday with non-standard week numbering.

or

ISO1 through ISO7, representing Sunday through Saturday with ISO standard week numbering.

Note: ISO is a synonym for ISO2.

The ISO standard establishes Monday as the first day of the week, so to be fully ISO compliant, the WEEKFIRST parameter should be set to ISO or ISO2.

Determining Collation of Data

You can use SET commands in a procedure to determine the collation criteria for sorted data. This parameter can also be set in the LOCALE settings on the Web Console to apply automatically to any procedures created on the Reporting Server for which this SET parameters is not defined. For more information, see [How to Change LOCALE Settings on the WebFOCUS Reporting Server](#) on page 63.

COLLATION

Set the COLLATION parameter in a procedure to override the COLLATION setting on the Reporting Server.

The syntax is:

```
SET COLLATION = {BINARY|SRV_CI|SRV_CS|CODEPAGE}
```

where:

BINARY

Bases the collation sequence on binary values.

SRV_CI

Bases collation sequence on the LANGUAGE setting, and is case-insensitive.

SRV_CS

Bases collation sequence on the LANGUAGE setting, and is case-sensitive.

CODEPAGE

Bases collation sequence on the code page in effect, and is case-sensitive. CODEPAGE is the default value.

In most cases, CODEPAGE is the same as BINARY. The only differences are for Danish, Finnish, German, Norwegian, and Swedish in an EBCDIC environment.

Displaying National Characters in Server-Side Graphics on UNIX and Linux

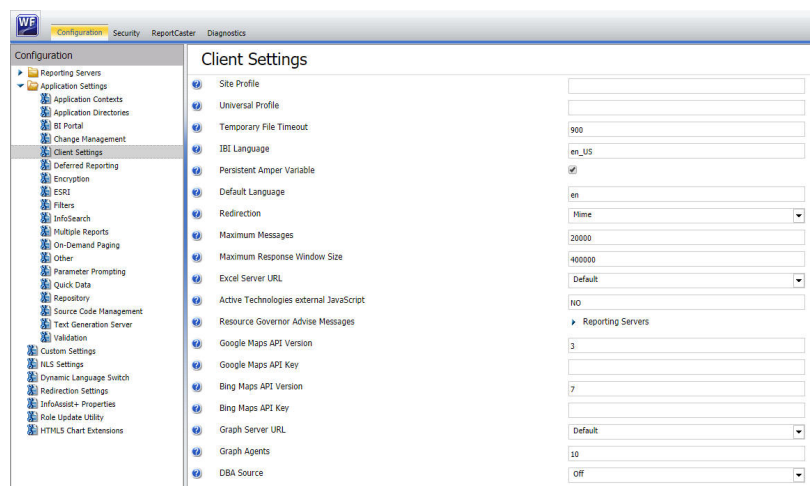
Procedure: How to Set the LANG Parameter in the Client Configuration File (cgivars.wfs)

1. Sign in to the WebFOCUS Administration Console.

The Configuration tab opens.

2. From the navigation pane, click *Client Settings*.

The Client Settings page appears, as shown in the following image.



3. Set the IBI Language parameter to the locale, as shown in the following image.



The following table displays some of the more common values for IBI Language.

Locale ID	Language and Country
zh_CN	Chinese (Simplified), China
zh_TW	Chinese (Traditional), Taiwan
da_DK	Danish, Denmark
de_DE	German, Germany

Locale ID	Language and Country
en_US	English, US
es_ES	Spanish, Spain
fr_FR	French, France
it_IT	Italian, Italy
ja_JP	Japanese, Japan
ko_KR	Korean, South Korea
pt_BR	Portuguese, Brazil

4. Click *Save* to store the settings.
5. Click *Close* to exit the console.

***Procedure:* How to Display UNIX and Linux Locale Values**

1. At the shell prompt, type:
2. A list of supported values displays.

```
locale -a
```

For example, on UNIX platforms configured for Japanese, the value can be any of the following depending on how the locale is configured on the web server.

- ☐ ja_JP
- ☐ ja_JP.eucjp
- ☐ ja_JP.ujis
- ☐ ja_JP.utf8
- ☐ japanese
- ☐ japanese.euc
- ☐ japanese.sjis

Specifying Multilingual Metadata in a Master File

The FILE declaration in a Master File can have a REMARKS or DESCRIPTION attribute that displays when browsing through Master Files using front-end tools, such as the App Studio Report canvas. Each FIELD declaration can have a DESCRIPTION attribute that provides documentation about that field and can be displayed instead of the field name in those tools. A FIELD declaration can also specify a TITLE attribute that provides a default column heading (title) in a report.

Master Files support descriptions in multiple languages. The description used depends on the value of the LANG parameter and whether a DESC_*ln* attribute is specified in the Master File, where *ln* identifies the language to which the description applies. In order to display these descriptions properly, all of the languages used must be consistent with your NLS configuration.

In a report, a column's heading is determined by:

1. A heading specified in the report request using the AS phrase.
2. A TITLE attribute in the Master File, if no AS phrase is specified in the request and SET TITLES=ON.
3. The field name specified in the Master File, if no AS phrase or TITLE attribute is specified, or if SET TITLES=OFF.

Syntax: **How to Specify Multilingual DESCRIPTION and TITLE Attributes**

For a file declaration in a Master File, use the following syntax:

```

FILE = filename,
.
.
.
{REMARKS|DESC} = default_desc
DESC_ln = desc_for_ln
.
.
.
FIELDNAME = field, ...
.
.
.
TITLE = default_column_heading
TITLE_ln = column_heading_for_ln
.
.
.
DESC = default_desc
DESC_ln = desc_for_ln
.
.
.

```

For a field declaration in a Master File, use the following syntax:

```

FIELDNAME = field, ...
.
.
.
TITLE = default_column_heading
TITLE_ln = column_heading_for_ln
.
.
.
DESC = default_desc
DESC_ln = desc_for_ln
.
.
.

```

where:

field

Is a field in the Master File.

default_column_heading

Is the column heading to use when SET TITLES=ON and either the LANG parameter is set to the default language for the server, or another language is set but the Master File has no corresponding TITLE_ln attribute for that field. This column heading is also used if an the ln value is invalid.

TITLE_1n

Specifies the language to which the associated column heading applies.

column_heading_for_1n

Specifies the text of the column heading for the specified language. That column heading is used when SET TITLES=ON, the LANG parameter is set to a non-default language for the server, and the Master File has a corresponding TITLE_1n attribute, where 1n is the two-digit code for the language specified by the LANG parameter. Valid values for 1n are the two-letter ISO 639 language code abbreviations.

default_desc

Is the description to use when either the LANG parameter is set to the default language for the server, or another language is set but the Master File has no corresponding DESC_1n attribute for that field. This description is also used if an the 1n value is invalid.

DESC_1n

Specifies the language to which the associated description applies.

desc_for_1n

Specifies the description text for the specified language. This description is used when the LANG parameter is set to a non-default language for the server and the Master File has a corresponding DESC_1n attribute. Valid values for 1n are two-letter ISO 639 language code abbreviations, as shown in the following table.

Language Name	Two-Letter Language Code	Three-Letter Language Abbreviation
Arabic	ar	ARB
Baltic	lt, lv	BAL
Chinese - Simplified GB	zh	PRC
Chinese - Traditional Big-5	tw	ROC
Czech	cs	CZE
Danish	da	DAN
Dutch	nl	DUT
English - American	en	AME or ENG

Language Name	Two-Letter Language Code	Three-Letter Language Abbreviation
English - UK	uk	UKE
Finnish	fi	FIN
French - Canadian	fc	FRE
French - Standard	fr	FRE
German - Austrian	at	GER
German - Standard	de	GER
Greek	el	GRE
Hebrew	iw	HEW
Italian	it	ITA
Japanese - Shift-JIS(cp942) on ascii cp939 on EBCDIC	ja	JPN
Japanese - EUC(cp10942) on ascii (UNIX)	je	JPE
Korean	ko	KOR
Norwegian	no	NOR
Polish	pl	POL
Portuguese - Brazilian	br	POR
Portuguese - Portugal	pt	POR
Russian	ru	RUS
Spanish	es	SPA
Swedish	sv	SWE
Thai	th	THA

Language Name	Two-Letter Language Code	Three-Letter Language Abbreviation
Turkish	tr	TUR

Syntax: **How to Activate the Use of a Language**

Issue the SET LANG command in a supported profile, on the command line, or in a FOCEXEC to set the language of error messages when set for a Reporting Server and localize report titles if the Master File contains alternate language TITLE attributes.

For information on using SET LANG to localize a report, see [Creating Multi-Locale Applications](#) on page 221.

The syntax is as follows:

```
SET LANG = lng
```

or

```
SET LANG = ln
```

In the nlscfg.err NLSCGF ERRORS configuration file, issue the following command:

```
LANG = lng
```

where:

```
lng
```

Is the three-letter abbreviation for the language.

```
ln
```

Is the two-letter ISO language code.

Note: If SET LANG is used in a procedure, its value will override the values set in nlscfg.err or in any profile.

Reference: **Usage Notes for Multilingual Metadata**

- ❑ To generate the correct characters, all languages used must be on the code page specified at server startup. To change the code page, you must stop and restart the server with the new code page.
- ❑ Master Files should be stored using the server code page used by FOCUS.

- ❑ Multilingual descriptions are supported with all fields described in the Master File, including DEFINE and COMPUTE fields.
- ❑ The user must create the NLSCFG file. On z/OS, the NLSCFG file must be a member in the concatenation of data sets allocated to DDNAME ERRORS. On z/VM, it must have filetype ERRORS.
- ❑ If you issue a HOLD command, only one TITLE attribute is propagated to the HOLD Master File. Its value is the column heading that would have appeared on the report output.

Example: **Using Multilingual Descriptions in a Master File**

The following Master File for the CENTINV data source specifies French descriptions (DESC_FR) and Spanish descriptions (DESC_ES) as well as default descriptions (DESC) for the PROD_NUM and PRODNAME fields:

```
FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
    DESCRIPTION='Product Number'
    DESC='Product Number',
    DESC_ES='Numero de Producto',
    DESC_FR='Nombre de Produit', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
    WITHIN=PRODCAT,
    DESCRIPTION='Product Name'
    DESC_FR='Nom de Produit',
    DESC_ES='Nombre de Producto', $
  FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
    DESCRIPTION='Quantity In Stock', $
  FIELD=PRICE, ALIAS=RETAIL, FORMAT=D10.2,
    TITLE='Price:',
    DESCRIPTION=Price, $
```

Example: **Using Multilingual Titles in a Request**

The following Master File for the CENTINV data source specifies French titles (TITLE_FR) and Spanish titles (TITLE_ES) as well as default titles (TITLE) for the PROD_NUM and PRODNAME fields:

```

FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
  TITLE='Product,Number:',
  TITLE_FR='Nombre,de Produit:',
  TITLE_ES='Numero,de Producto:',
  DESCRIPTION='Product Number', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
  WITHIN=PRODCAT,
  TITLE='Product,Name:',
  TITLE_FR='Nom,de Produit:',
  TITLE_ES='Nombre,de Producto:'
  DESCRIPTION='Product Name', $
  FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
  TITLE='Quantity,In Stock:',
  DESCRIPTION='Quantity In Stock', $
  FIELD=PRICE, ALIAS=RETAIL, FORMAT=D10.2,
  TITLE='Price:',
  DESCRIPTION=Price, $

```

The default language for the server code page is English and, by default, SET TITLES=ON. Therefore, the following request, uses the TITLE attributes to produce column headings that are all in English:

```

TABLE FILE CENTINV
PRINT PROD_NUM PRODNAME PRICE
WHERE PRICE LT 200
END

```

The output is:

Product Number:	Product Name:	Price:
-----	-----	-----
1004	2 Hd VCR LCD Menu	179.00
1008	DVD Upgrade Unit for Cent. VCR	199.00
1026	AR3 35MM Camera 10 X	129.00
1028	AR2 35MM Camera 8 X	109.00
1030	QX Portable CD Player	169.00
1032	R5 Micro Digital Tape Recorder	89.00

Now, issue either of the following commands to set the language to Spanish, and run the same request:

☐ SET LANG = SPA

☐ SET LANG = ES

The output now displays column headings from the TITLE_ES attributes where they exist (Product Number and Product Name). Where no Spanish title is specified (the Price field), the column heading in the TITLE attribute appears:

Numero de Producto:	Nombre de Producto:	Price:
-----	-----	-----
1004	2 Hd VCR LCD Menu	179.00
1008	DVD Upgrade Unit for Cent. VCR	199.00
1026	AR3 35MM Camera 10 X	129.00
1028	AR2 35MM Camera 8 X	109.00
1030	QX Portable CD Player	169.00
1032	R5 Micro Digital Tape Recorder	89.00

Storing Localized Metadata in Language Files

If you want to centralize localized column titles, descriptions, and prompts, and apply them to multiple Master Files, you can create a set of translation files and use the TRANS_FILE attribute in a Master File to invoke them. You can set up the files totally manually, or you can use the LNGPREP utility to prepare the files for translation.

LNGPREP Utility: Preparing Metadata Language Files

The LNGPREP utility extracts TITLE, DESCRIPTION, CAPTION, and PROMPT attribute values from application Master Files into specially formatted language translation files for each language you need. Once you have the contents of these language files translated, your users can run these applications in the language they select.

LNGPREP does two things. It extracts attribute values from a Master File into language files, and it inserts or updates the TRANS_FILE attribute in the Master File with a value identifying the application folder where the language files reside and a prefix used for naming the set of language files. If the Master File is part of a cluster, LNGPREP will extract translatable strings from every Master File referenced in the cluster, and will update each with the same TRANS_FILE value.

LNGPREP requires an input file listing the three-character codes of the languages you need.

The name of each language file starts with the prefix specified in the TRANS_FILE value, followed by a three-character language code, and the extension *.lng*.

For example, assume the language input file contains the French and Spanish language codes:

```
fre
spa
```

If the Master File specifies:

```
trans_file = xlate/xl_
```

The language translation files would be in the *xlate* application folder, named:

❑ *xl_fre.lng* for French.

❑ *xl_spa.lng* for Spanish.

Reference: The Base Language File

Each Master File must have a single base language in which the DESCRIPTION, TITLE, CAPTION, and PROMPT attributes are specified. This language does not need to be English.

LNGPREP extracts these attribute values into the base language file, whose language code, for historical reasons, is *eng*. In this case, *eng* does not mean English. It means whatever language the Master File is written in.

The base language file (*prefixeng.lng*) should never be hand edited. All other lng files must be hand edited by translators, to translate the string values from the base language to the appropriate language.

Translating Applications into English

Since language code *eng* is reserved to mean base language, you cannot use it to contain English translations of an application whose base language is not English. In those cases, use any of the other English dialect language codes, such as AME, UKE, CAE, or AUE. For example, if the base language is German, specify AME in the languages file, run LNGPREP, and it will produce *prefixeng.lng* and *prefixame.lng* files, both in German. Translate the contents of *prefixame.lng* into English. Leave *prefixeng.lng* untouched.

Reference: How Translated Master File Attributes Display

Each language file contains a line for each attribute value from a related set of Master Files. Each attribute value has a unique index number assigned to it. For example, if the Master File contains FIELDNAME=PRODUCT_CATEGORY, TITLE='Product,Category', and that TITLE happens to be the 39th translatable attribute value, LNGPREP will produce lng files all containing the line:

```
39 = Product,Category
```

Your French translator will edit *prefixfre.lng*, leaving the index values unchanged while translating the string values, producing, in this case,

```
39 = Produit,Catégorie
```

At run time, when the TITLE for field PRODUCT_CATEGORY needs to be displayed, if WebFOCUS is configured for LANG=FRE, WebFOCUS looks up "Product,Category" in *prefixeng.Ing*, finds index value 39, looks up 39 in *prefixfre.Ing*, and displays the TITLE as "Produit,Catégorie."

LNGPREP Modes

You can run LNGPREP from the Web Console using the Prepare Translation Files option, or you can run it using syntax. In either case, you must first create a configuration file containing the three-character language codes for each translation file you need, one language code on each line. The first invocation of LNGPREP for a given Master File adds the TRANS_FILE attribute in that and all related Master Files, creates the base language file by scanning the Master Files for supported attribute values, and creates a copy of the base language file with the correct name for each additional language. Then, a translator has to translate the values in each additional language file from the base language to the correct language for that file.

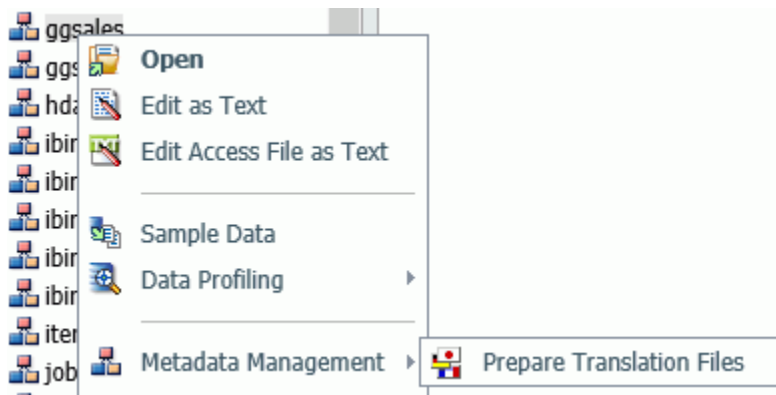
On each subsequent run, LNGPREP will check for updates to the list of related Master Files and attribute values and update the files as needed. Translators will then have to translate any attribute values added to the language files.

Reference: LNGPREP Best Practice

The recommended best practice is to create an app directory solely for the purpose of containing .Ing files, and use this appname and a common prefix value for all LNGPREP commands. In addition, put the languages *fn.cfg* file in this app folder. This will create one set of .Ing files for all apps, minimizing the time and effort spent on translation.

Procedure: How to Prepare Metadata Language Files Using the Web Console

1. Right-click a synonym, point to *Metadata Management*, then click *Prepare Translation Files*, as shown in the following image.



The Set Translation Files page opens, as shown in the following image.



Set Translation Files for ibisamp/ggsales.mas

Application for Translation Files: 

Prefix:

Languages File: 

2. Enter the following values or accept the defaults.

Application for Translation Files

Is the name of the application where the language files will be stored. You can click the ellipsis to select an application from the current application path. By default, it is the application where the synonym resides.

Prefix

Is the prefix value for the translation files for the selected synonym.

Languages File

Is the file containing the list of language codes for which translation files should be prepared. The file must have the extension .cfg, be stored in an application directory on the application path, and have one language code on each line. You can click the ellipsis to select the application where the languages file is stored.

3. Click OK.

The language files are prepared using the application, prefix, and languages configuration file you specified. A status page will open listing the language files created and the Master Files processed.

Syntax: How to Run the LNGPREP Command Using Syntax

```
LNGPREP FILE n_part_name LNGAPP appname LNGPREFIX prefix  
          LNGFILE appname/fn
```

where:

n_part_name

Specifies the n-part (app1/app2...) name of a Master File.

appname

Specifies the location where .lng files will be written and updated.

prefix

Specifies the literal characters that will precede the three-character language code in the names of the .lng files.

appname/fn

Specifies the *appname* and *filename* of a user-created .cfg file containing the list of three-character language codes, one per line. For example, the following file named langretail.cfg contains language codes for American English, French, and Japanese:

```
ame
fre
jpn
```

Example: Sample LNGPREP Command

Assume the lnglist.cfg file contains the language codes fre (French) and spa (Spanish):

```
fre
spa
```

Issue the following LNGPREP command:

```
LNGPREP FILE weather/forecast LNGAPP xlate LNGPREFIX tq_ LNGFILE xlate/lnglist
```

Alternately, you can right-click the forecast synonym, point to *Metadata Management*, and select *Prepare Translation Files*. The Set Translation File for weather/forecast.mas page opens, as shown in the following image. Enter the values shown in the following image and click OK.

Set Translation Files for weather/forecast.mas


Application for Translation Files: xlate

Prefix: tq_

Languages File: xlate/lnglist.cfg

OK Cancel

The following language files will be created:

- ☐ xlate/tq_eng.lng
- ☐ xlate/tq_fre.lng
- ☐ xlate/tq_spa.lng

The Master File weather/forecast.mas will be updated with the following attribute:

```
TRANS_FILE= xlate/tq_
```

Translators then have to translate the values in `xlate/tq_fre.lng` and `xlate/tq_spa.lng`.

Using LNGPREP for a Multi-Locale Application

For an example of how you can use LNGPREP to create a single application that can be run in multiple languages, see [Creating a Multi-Locale Application Using LNGPREP](#) on page 221.

Adding the Code Page Parameter to a Master File

You can place a parameter in a Master File to document the code page of the associated data source, for example, if your data source is on a database that uses a different code page than your WebFOCUS Reporting Server.

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

Procedure: How to Add the Code Page Parameter to a Master File Using the Synonym Editor

You can set the code page for a Master File by modifying the CODEPAGE parameter in the Properties panel in the Synonym Editor in App Studio or the Data Management Console (DMC).

1. Open a synonym in the Synonym Editor.
 - ❑ **App Studio.** In the Environments Tree panel, expand the *Data Servers* node, the server, and the application folder containing your data. Optionally, use the Environments Tree toolbar to filter for Synonyms. Right-click the Master File you wish to open and click *Open*.
 - ❑ **Data Management Console.** In the navigation pane, expand the *LOOPBACK* server node, *Application Directories*, and the application folder containing your synonyms. Optionally, click *Synonyms* in the Filter group on the ribbon to show only synonyms in the navigation pane. Right-click the synonym whose Master File you wish to open and click *Open*.

The synonym opens in the Synonym Editor.

2. Make sure the Field View tab is displayed in the Synonym Editor. Right-click the name of the synonym in the Display Name (Title) column (*application/filename*) and click *Properties*. The Properties panel opens.
3. In the Properties panel, select a code page from the CODEPAGE drop-down menu. The code page will be applied to the Master File of the synonym.

Procedure: How to Add the Code Page Parameter to a Master File Using the Text Editor

Alternatively, you can manually add the code page definition to the Master File using the text editor in App Studio, the Data Management Console, or the Web Console.

1. Open a Master File in the text editor.

- ☐ **App Studio.** In the Environments Tree panel, expand the *Data Servers* node, the server, and the application folder containing your data. Optionally, use the Environments Tree toolbar to filter for Synonyms. Right-click the Master File you wish to open and click *Open in Text Editor*.
- ☐ **Data Management Console.** In the navigation pane, expand the *LOOPBACK* server node, *Application Directories*, and the application folder containing your synonyms. Optionally, click *Synonyms* in the Filter group on the ribbon to show only synonyms in the navigation pane. Right-click the synonym whose Master File you wish to open and click *Edit as Text*.
- ☐ **Web Console.** In the Web Console, click the *Applications* tab. In the navigation pane, expand the application folder that contains your synonym. Optionally, click *Filter* in the Application Preferences group on the ribbon, point to *Synonyms*, and click *All Synonyms* to show only synonyms in the navigation pane. Right-click the synonym whose Master File you wish to edit and click *Edit as Text*.

The Master File opens in the text editor.

2. Determine the code page you would like to apply.

A complete list of code pages is available in [Language and Default Code Pages for Windows, UNIX, and Linux](#) on page 74.

3. On the first line, after the SUFFIX parameter, type `CODEPAGE=` followed by the number of the code page that you want to apply and a comma.

Example: Adding the Code Page Parameter to a Master File

In this example, the user added code page documentation to the EMPLOYEE Master File. The selected code page value, 863 (IBM PC Canadian French), appears on line 5 in the text editor, on the first line of the Master File (each line ends with a dollar sign).

```

1  $-----$
2  $ Copyright (c) Information Builders, Inc. All rights reserved. @MFSM_NOPROLOG@ $
3  $-----$
4  FILENAME=EMPLOYEE, SUFFIX=FOC ,
5  CODEPAGE=863,
6  REMARKS='Legacy Metadata Sample: employee', $
7  SEGMENT=EMPINFO, SEGTYPE=S1, $
8  FIELDNAME=EMP_ID, ALIAS=EID, USAGE=A9, $
9  FIELDNAME=LAST_NAME, ALIAS=LN, USAGE=A15, $
10 FIELDNAME=FIRST_NAME, ALIAS=FN, USAGE=A10, $

```

Running PDF or PS Reports with International Fonts in App Studio

Some international languages such as Chinese, Hebrew, and Thai require specific type-1 fonts in order to display NLS characters correctly in a PDF or PS report. The WebFOCUS Reporting Server uses the LANG=XXX value to read the pdfXXX.fmp file instead of the default pdf.fmp file so that the proper international PDF font that supports the NLS characters is used.

However, when you execute a procedure from an App Studio canvas, StyleSheet information is added to the request and the default font name in the associated StyleSheet (*.sty) file is used. The FONT attribute from the default StyleSheet overrides the PDF font information from the WebFOCUS Reporting Server, and therefore NLS characters may not display correctly. This can be corrected by changing the font from the App Studio tool, or editing the StyleSheet file.

Additionally, the embedded PDF viewer for a browser may not display NLS characters correctly. If the StyleSheet is already set to use an NLS compatible font, but NLS characters still do not display correctly, you can use font embedding with the fontuser.xml file, as described in [Unicode PDF Output](#) on page 104. Alternatively, you can configure your browser to use Adobe Reader instead of the native PDF viewer.

Procedure: How to Display NLS Characters Correctly in a PDF or PS Report in App Studio

To display NLS characters correctly in PDF or PS reports you execute from App Studio, you must edit the font attribute in the StyleSheet file, or change the font directly from the App Studio tool.

To change the font for a StyleSheet file (affects all reports that use the StyleSheet):

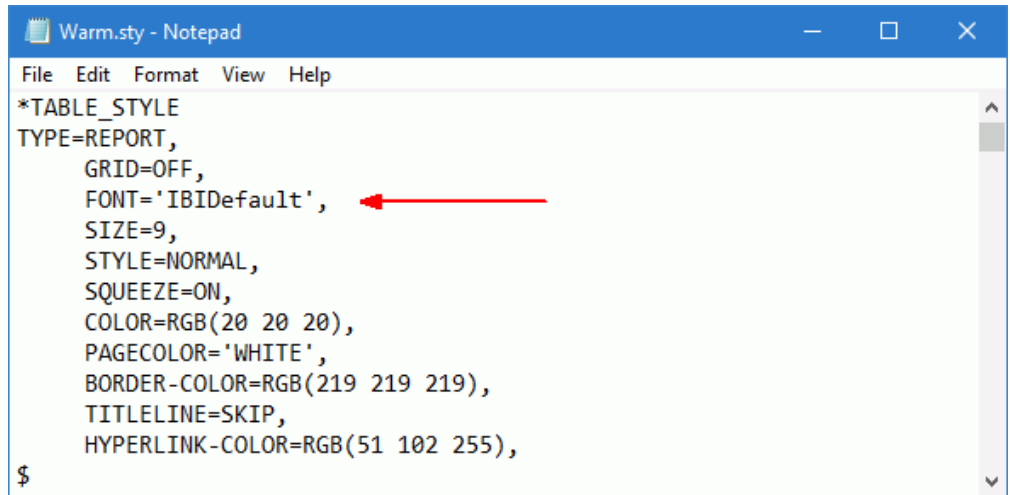
1. Open the associated StyleSheet in a text editor.

Unless you have selected a specific StyleSheet for a report, the default StyleSheet is always warm.sty.

Note: StyleSheet files (*.sty) are located in the *drive:\ibi\WebFOCUSnn\ibi_html\ibi_themes* and *drive:\ibi\AppStudio\ibi_html\ibi_themes* directories, where *nn* is your WebFOCUS release number.

2. Find and change the "FONT=" line.

The following image shows the default StyleSheet file (warm.sty) and the FONT attribute.



Procedure: How to Change the Font for a Specific Report in App Studio

On the Report tab, in the Style group, with the Scope set to *Report*, change the font. The report should update immediately on the canvas with the selected font.

Helpful NLS Functions

Functions operate on one or more arguments and return a single value. The returned value can be stored in a field, assigned to a Dialogue Manager variable, used in a calculation or other processing, or used in a selection or validation test. Functions provide a convenient way to perform certain calculations and manipulations.

You can use these functions to transform characters in order to facilitate translation between languages and code pages, or you can use them in DEFINE or COMPUTE fields to modify your data within a procedure. For more information, see the *Using Functions* and *Functions Reference* manuals.

In this appendix:

- ❑ [BYTVAL: Translating a Character to Decimal](#)
- ❑ [CHAR: Returning a Character Based on a Numeric Code](#)
- ❑ [CTRAN: Translating One Character to Another](#)
- ❑ [DCTRAN: Translating A Single-Byte or Double-Byte Character to Another](#)
- ❑ [DATECVT: Converting the Format of a Date](#)
- ❑ [DATETRAN: Formatting Dates in International Formats](#)
- ❑ [HDATE: Converting the Date Portion of a Date-Time Value to a Date Format](#)
- ❑ [HEXBYT: Converting a Decimal Integer to a Character](#)
- ❑ [HEXTYPE: Returning the Hexadecimal View of an Input Value](#)
- ❑ [LCWORD2: Converting a String to Mixed-Case](#)
- ❑ [LCWORD3: Converting a String to Mixed-Case](#)
- ❑ [LOCASE: Converting Text to Lowercase](#)
- ❑ [PATTERN: Generating a Pattern From a String](#)
- ❑ [REPLACE: Replacing a String](#)
- ❑ [REVERSE: Reversing the Characters in a String](#)
- ❑ [STRIP: Removing a Character From a String](#)
- ❑ [DSTRIP: Removing a Single-Byte or Double-Byte Character From a String](#)
- ❑ [SFTDEL: Deleting the Shift Code From DBCS Data](#)
- ❑ [SFTINS: Inserting the Shift Code Into DBCS Data](#)
- ❑ [STRREP: Replacing Character Strings](#)
- ❑ [UFMT: Converting an Alphanumeric String to Hexadecimal](#)

- ☐ JPTRANS: Converting Japanese Specific Characters
 - ☐ KKFCUT: Truncating a String
 - ☐ LCWORD: Converting a String to Mixed-Case
 - ☐ UPCASE: Converting Text to Uppercase
-

BYTVAL: Translating a Character to Decimal

Available Languages: reporting, Maintain

The BYTVAL function translates a character to the ASCII, EBCDIC, or Unicode decimal value that represents it, depending on the operating system.

Syntax: How to Translate a Character

`BYTVAL(character, output)`

where:

character

Alphanumeric

Is the character to be translated. You can specify a field or variable that contains the character, or the character itself enclosed in single quotation marks. If you supply more than one character, the function evaluates the first.

output

Integer

Is the name of the field that contains the corresponding decimal value, or the format of the output value enclosed in single quotation marks.

Example: Translating the First Character of a Field

BYTVAL translates the first character of LAST_NAME into its ASCII or EBCDIC decimal value and stores the result in LAST_INIT_CODE. Since the input string has more than one character, BYTVAL evaluates the first one.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND
COMPUTE LAST_INIT_CODE/I3 = BYTVAL(LAST_NAME, 'I3');
WHERE DEPARTMENT EQ 'MIS';
END
```

The output on an ASCII platform is:

LAST_NAME	LAST_INIT_CODE
SMITH	83
JONES	74
MCCOY	77
BLACKWOOD	66
GREENSPAN	71
CROSS	67

The output on an EBCDIC platform is:

LAST_NAME	LAST_INIT_CODE
SMITH	226
JONES	209
MCCOY	212
BLACKWOOD	194
GREENSPAN	199
CROSS	195

Example: Returning the EBCDIC Value With Dialogue Manager

This Dialogue Manager request prompts for a character, then returns the corresponding number. The following reflects the results on the Windows platform.

```
-SET &CODE = BYTVAL(&CHAR, 'I3');
-HTMLFORM BEGIN
<HTML>
<BODY>
THE EQUIVALENT VALUE IS &CODE
</BODY>
</HTML>
-HTMLFORM END
```

Assume the value entered for &CHAR is an exclamation point (!). The output is:

```
THE EQUIVALENT VALUE IS 33
```

CHAR: Returning a Character Based on a Numeric Code

The CHAR function accepts a decimal integer and returns the character identified by that number converted to ASCII or EBCDIC, depending on the operating environment. The output is returned as variable length alphanumeric. If the number is above the range of valid characters, a null value is returned.

Syntax: How to Return a Character Based on a Numeric Code

CHAR(number_code)

where:

number_code

Integer

Is a field, number, or numeric expression whose whole absolute value will be used as a number code to retrieve an output character.

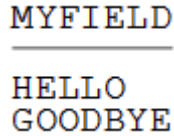
For example, a TAB character is returned by CHAR(9) in ASCII environments, or by CHAR(5) in EBCDIC environments.

Example: Using the CHAR Function to Insert Control Characters Into a String

The following request defines a field with carriage return (CHAR(13)) and line feed (CHAR(10)) characters inserted between the words HELLO and GOODBYE (in an ASCII environment). To show that these characters were inserted, the output is generated in PDF format and the StyleSheet attribute LINEBREAK='CRLF' is used to have these characters respected and print the field value on two lines.

```
DEFINE FILE WF_RETAIL_LITE
MYFIELD/A20 WITH COUNTRY_NAME='HELLO' | CHAR(13) | CHAR(10) | 'GOODBYE';
END
TABLE FILE WF_RETAIL_LITE
SUM MYFIELD
ON TABLE PCHOLD FORMAT PDF
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
TYPE=REPORT,LINEBREAK='CRLF', $
ENDSTYLE
END
```

The output is shown in the following image.



```
MYFIELD
-----
HELLO
GOODBYE
```

CTRAN: Translating One Character to Another

Available Languages: reporting, Maintain

The CTRAN function translates a character within a character string to another character based on its decimal value. This function is especially useful for changing replacement characters to unavailable characters, or to characters that are difficult to input or unavailable on your keyboard. It can also be used for inputting characters that are difficult to enter when responding to a Dialogue Manager -PROMPT command, such as a comma or apostrophe. It eliminates the need to enclose entries in single quotation marks (').

To use CTRAN, you must know the decimal equivalent of the characters in internal machine representation. Note that the coding chart for conversion is platform dependent, hence your platform and configuration option determines whether ASCII, EBCDIC, or Unicode coding is used.

In Unicode configurations, this function uses values in the range:

- ❑ 0 to 255 for 1-byte characters.
- ❑ 256 to 65535 for 2-byte characters.
- ❑ 65536 to 16777215 for 3-byte characters.
- ❑ 16777216 to 4294967295 for 4-byte characters (primarily for EBCDIC).

Syntax: How to Translate One Character to Another

```
CTRAN(length, source_string, decimal, decvalue, output)
```

where:

length

Integer

Is the number of characters in the source string, or a field that contains the length.

source_string

Alphanumeric

Is the character string to be translated enclosed in single quotation marks ('), or the field or variable that contains the character string.

decimal

Integer

Is the ASCII or EBCDIC decimal value of the character to be translated.

decvalue

Integer

Is the ASCII or EBCDIC decimal value of the character to be used as a substitute for *decimal*.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks.

Example: Translating Spaces to Underscores on an ASCII Platform

CTRAN translates the spaces in ADDRESS_LN3 (ASCII decimal value 32) to underscores (ASCII decimal value 95), and stores the result in ALT_ADDR:

```
TABLE FILE EMPLOYEE
PRINT ADDRESS_LN3 AND COMPUTE
ALT_ADDR/A20 = CTRAN(20, ADDRESS_LN3, 32, 95, ALT_ADDR);
BY EMP_ID
WHERE TYPE EQ 'HSM';
END
```

The output is:

EMP_ID	ADDRESS_LN3	ALT_ADDR
-----	-----	-----
117593129	RUTHERFORD NJ 07073	RUTHERFORD_NJ_07073__
119265415	NEW YORK NY 10039	NEW_YORK_NY_10039__
119329144	FREEPORT NY 11520	FREEPORT_NY_11520__
123764317	NEW YORK NY 10001	NEW_YORK_NY_10001__
126724188	FREEPORT NY 11520	FREEPORT_NY_11520__
451123478	ROSELAND NJ 07068	ROSELAND_NJ_07068__
543729165	JERSEY CITY NJ 07300	JERSEY_CITY_NJ_07300
818692173	FLUSHING NY 11354	FLUSHING_NY_11354

Example: Translating Spaces to Underscores on an EBCDIC Platform

CTRAN translates the spaces in ADDRESS_LN3 (EBCDIC decimal value 64) to underscores (EBCDIC decimal value 109) and stores the result in ALT_ADDR:

```
TABLE FILE EMPLOYEE
PRINT ADDRESS_LN3 AND COMPUTE
ALT_ADDR/A20 = CTRAN(20, ADDRESS_LN3, 64, 109, ALT_ADDR) ;
BY EMP_ID
WHERE TYPE EQ 'HSM'
END
```

The output is:

EMP_ID	ADDRESS_LN3	ALT_ADDR
-----	-----	-----
117593129	RUTHERFORD NJ 07073	RUTHERFORD_NJ_07073_
119265415	NEW YORK NY 10039	NEW_YORK_NY_10039__
119329144	FREEPORT NY 11520	FREEPORT_NY_11520___
123764317	NEW YORK NY 10001	NEW_YORK_NY_10001__
126724188	FREEPORT NY 11520	FREEPORT_NY_11520___
451123478	ROSELAND NJ 07068	ROSELAND_NJ_07068__
543729165	JERSEY CITY NJ 07300	JERSEY_CITY_NJ_07300
818692173	FLUSHING NY 11354	FLUSHING_NY_11354___

DCTRAN: Translating A Single-Byte or Double-Byte Character to Another

The DCTRAN function translates a single-byte or double-byte character within a character string to another character based on its decimal value. To use DCTRAN, you need to know the decimal equivalent of the characters in internal machine representation.

The DCTRAN function can translate single-byte to double-byte characters and double-byte to single-byte characters, as well as single-byte to single-byte characters and double-byte to double-byte characters.

Syntax: How to Translate a Single-Byte or Double-Byte Character to Another

```
DCTRAN(length, source_string, indecimal, outdecimal, output)
```

where:

length

Double

Is the number of characters in *source_string*.

source_string

Alphanumeric

Is the character string to be translated.

indecimal

Double

Is the ASCII or EBCDIC decimal value of the character to be translated.

outdecimal

Double

Is the ASCII or EBCDIC decimal value of the character to be used as a substitute for *indecimal*.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks (').

Example: Using DCTRAN to Translate Double-Byte Characters

In the following:

```
DCTRAN(8, 'A/A本B語', 177, 70, A8)
```

For A/A本B語, the result is AFA本B語.

DATECVT: Converting the Format of a Date

Available Languages: reporting, Maintain

The DATECVT function converts the field value of any standard date format or legacy date format into a date format (offset from the base date), in the desired standard date format or legacy date format. If you supply an invalid format, DATECVT returns a zero or a blank.

DATECVT turns off optimization and compilation.

Note: You can use simple assignment instead of calling this function.

Syntax: **How to Convert a Date Format**

```
DATECVT(date, 'in_format', output)
```

where:

date

Date

Is the date to be converted. If you supply an invalid date, DATECVT returns zero. When the conversion is performed, a legacy date obeys any DEFCENT and YRTHRESH parameter settings supplied for that field.

in_format

Alphanumeric

Is the format of the date enclosed in single quotation marks. It is one of the following:

- ❑ A non-legacy date format (for example, YYMD, YQ, M, DMY, JUL).
- ❑ A legacy date format (for example, I6YMD or A8MDYY).
- ❑ A non-date format (such as I8 or A6). A non-date format in *in_format* functions as an offset from the base date of a YYMD field (12/31/1900).

output

Alphanumeric

Is the output format enclosed in single quotation marks or a field containing the format. It is one of the following:

- ❑ A non-legacy date format (for example, YYMD, YQ, M, DMY, JUL).
- ❑ A legacy date format (for example, I6YMD or A8MDYY).
- ❑ A non-date format (such as I8 or A6). This format type causes DATECVT to convert the date into a full component date and return it as a whole number in the format provided.

Example: **Converting a YYMD Date to DMY**

DATECVT converts 19991231 to 311299 and stores the result in CONV_FIELD:

```
CONV_FIELD/DMY = DATECVT(19991231, 'I8YYMD', 'DMY');
```

or

```
ONV_FIELD/DMY = DATECVT('19991231', 'A8YYMD', 'DMY');
```

Example: **Converting a Legacy Date to Date Format (Reporting)**

DATECVT converts HIRE_DATE from I6YMD legacy date format to YYMD date format:

```
TABLE FILE EMPLOYEE
PRINT FIRST_NAME AND HIRE_DATE AND COMPUTE
NEW_HIRE_DATE/YYMD = DATECVT(HIRE_DATE, 'I6YMD', 'YYMD');
BY LAST_NAME
WHERE DEPARTMENT EQ 'MIS';
END
```

The output is:

LAST_NAME	FIRST_NAME	HIRE_DATE	NEW_HIRE_DATE
-----	-----	-----	-----
BLACKWOOD	ROSEMARIE	82/04/01	1982/04/01
CROSS	BARBARA	81/11/02	1981/11/02
GREENSPAN	MARY	82/04/01	1982/04/01
JONES	DIANE	82/05/01	1982/05/01
MCCOY	JOHN	81/07/01	1981/07/01
SMITH	MARY	81/07/01	1981/07/01

DATETRAN: Formatting Dates in International Formats

Available Languages: reporting, Maintain

The DATETRAN function formats dates in international formats.

Syntax: **How to Format Dates in International Formats**

```
DATETRAN (indate, '(intype)', '([formatops]','lang', outlen, output)
```

where:

indate

Is the input date (in date format) to be formatted. Note that the date format cannot be an alphanumeric or numeric format with date display options (legacy date format).

intype

Is one of the following character strings indicating the input date components and the order in which you want them to display, enclosed in parentheses and single quotation marks.

The following table shows the single component input types:

Single Component Input Type	Description
' (W) '	Day of week component only (original format must have only W component).
' (M) '	Month component only (original format must have only M component).

The following table shows the two-component input types:

Two-Component Input Type	Description
' (YYM) '	Four-digit year followed by month.
' (YM) '	Two-digit year followed by month.
' (MY) '	Month component followed by four-digit year.
' (MY) '	Month component followed by two-digit year.

The following table shows the three-component input types:

Three-Component Input Type	Description
' (YYMD) '	Four-digit year followed by month followed by day.
' (YMD) '	Two-digit year followed by month followed by day.
' (DMYY) '	Day component followed by month followed by four-digit year.

Three-Component Input Type	Description
' (DMY) '	Day component followed by month followed by two-digit year.
' (MDYY) '	Month component followed by day followed by four-digit year.
' (MDY) '	Month component followed by day followed by two-digit year.
' (MD) '	Month component followed by day (derived from three-component date by ignoring year component).
' (DM) '	Day component followed by month (derived from three-component date by ignoring year component).

formatops

Is a string of zero or more formatting options enclosed in parentheses and single quotation marks. The parentheses and quotation marks are required even if you do not specify formatting options. Formatting options fall into the following categories:

- ☐ Options for suppressing initial zeros in month or day numbers.
Note: Zero suppression replaces initial zeros with blanks spaces.
- ☐ Options for translating month or day components to full or abbreviated uppercase or default case (mixed-case or lowercase depending on the language) names.
- ☐ Date delimiter options and options for punctuating a date with commas.

Valid options for suppressing initial zeros in month or day numbers are listed in the following table. Note that the initial zero is replaced by a blank space:

Format Option	Description
m	Zero-suppresses months (displays numeric months before October as 1 through 9 rather than 01 through 09).

Format Option	Description
<code>d</code>	Displays days before the tenth of the month as 1 through 9 rather than 01 through 09.
<code>dp</code>	Displays days before the tenth of the month as 1 through 9 rather than 01 through 09 with a period after the number.
<code>do</code>	Displays days before the tenth of the month as 1 through 9. For English (langcode EN) only, displays an ordinal suffix (st, nd, rd, or th) after the number.

The following table shows valid month and day name translation options:

Format Option	Description
<code>T</code>	Displays month as an abbreviated name, with no punctuation, all uppercase.
<code>TR</code>	Displays month as a full name, all uppercase.
<code>Tp</code>	Displays month as an abbreviated name, followed by a period, all uppercase.
<code>t</code>	Displays month as an abbreviated name with no punctuation. The name is all lowercase or initial uppercase, depending on language code.
<code>tr</code>	Displays month as a full name. The name is all lowercase or initial uppercase, depending on language code.
<code>tp</code>	Displays month as an abbreviated name, followed by a period. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).

Format Option	Description
<code>W</code>	Includes an abbreviated day-of-the-week name at the start of the displayed date, all uppercase with no punctuation.
<code>WR</code>	Includes a full day-of-the-week name at the start of the displayed date, all uppercase.
<code>Wp</code>	Includes an abbreviated day-of-the-week name at the start of the displayed date, all uppercase, followed by a period.
<code>w</code>	Includes an abbreviated day-of-the-week name at the start of the displayed date with no punctuation. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
<code>wr</code>	Includes a full day-of-the-week name at the start of the displayed date. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
<code>wP</code>	Includes an abbreviated day-of-the-week name at the start of the displayed date followed by a period. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
<code>X</code>	Includes an abbreviated day-of-the-week name at the end of the displayed date, all uppercase with no punctuation.
<code>XR</code>	Includes a full day-of-the-week name at the end of the displayed date, all uppercase.

Format Option	Description
<code>xp</code>	Includes an abbreviated day-of-the-week name at the end of the displayed date, all uppercase, followed by a period.
<code>x</code>	Includes an abbreviated day-of-the-week name at the end of the displayed date with no punctuation. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
<code>xr</code>	Includes a full day-of-the-week name at the end of the displayed date. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).
<code>xp</code>	Includes an abbreviated day-of-the-week name at the end of the displayed date followed by a period. The name displays in the default case of the specified language (for example, all lowercase for French and Spanish, initial uppercase for English and German).

The following table shows valid date delimiter options:

Format Option	Description
<code>B</code>	Uses a blank as the component delimiter. This is the default if the month or day of week is translated or if comma is used.
<code>.</code>	Uses a period (.) as the component delimiter.
<code>-</code>	Uses a minus sign (-) as the component delimiter. This is the default when the conditions for a blank default delimiter are not satisfied.

Format Option	Description
/	Uses a slash (/) as the component delimiter.
	Omits component delimiters.
K	Uses appropriate Asian characters as component delimiters.
c	Places a comma (,) after the month name (following T, Tp, TR, t, tp, or tr). Places a comma and blank after the day name (following W, Wp, WR, w, wp, or wr). Places a comma and blank before the day name (following X, XR, x, or xr).
e	Displays the Spanish or Portuguese word de or DE between the day and month, and between the month and year. The case of the word de is determined by the case of the month name. If the month is displayed in uppercase, DE is displayed. Otherwise, de is displayed. Useful for formats DMY, DMYy, MY, and MYy.
D	Inserts a comma (,) after the day number and before the general delimiter character specified.
Y	Inserts a comma (,) after the year and before the general delimiter character specified.

lang

Is the two-character standard ISO code for the language into which the date should be translated, enclosed in single quotation marks ('). Valid language codes are:

- ☐ 'AR' Arabic
- ☐ 'CS' Czech
- ☐ 'DA' Danish
- ☐ 'DE' German

- ☐ 'EN' English
- ☐ 'ES' Spanish
- ☐ 'FI' Finnish
- ☐ 'FR' French
- ☐ 'EL' Greek
- ☐ 'IW' Hebrew
- ☐ 'IT' Italian
- ☐ 'JA' Japanese
- ☐ 'KO' Korean
- ☐ 'LT' Lithuanian
- ☐ 'NL' Dutch
- ☐ 'NO' Norwegian
- ☐ 'PO' Polish
- ☐ 'PT' Portuguese
- ☐ 'RU' Russian
- ☐ 'SV' Swedish
- ☐ 'TH' Thai
- ☐ 'TR' Turkish
- ☐ 'TW' Chinese (Traditional)
- ☐ 'ZH' Chinese (Simplified)

outlen

Numeric

Is the length of the output field in bytes. If the length is insufficient, an all blank result is returned. If the length is greater than required, the field is padded with blanks on the right.

output

Alphanumeric

Is the name of the field that contains the translated date, or its format enclosed in single quotation marks.

Reference: Usage Notes for the DATETRAN Function

- ❑ The output field, though it must be type A, and not AnV, may in fact contain variable length information, since the lengths of month names and day names can vary, and also month and day numbers may be either one or two bytes long if a zero-suppression option is selected. Unused bytes are filled with blanks.
- ❑ All invalid and inconsistent inputs result in all blank output strings. Missing data also results in blank output.
- ❑ The base dates (1900-12-31 and 1900-12 or 1901-01) are treated as though the DATEDISPLAY setting were ON (that is, not automatically shown as blanks). To suppress the printing of base dates, which have an internal integer value of 0, test for 0 before calling DATETRAN. For example:

```
RESULT/A40 = IF DATE EQ 0 THEN ' ' ELSE  
              DATETRAN (DATE, '(YYMD)', '(.t)', 'FR', 40, 'A40');
```

- ❑ Valid translated date components are contained in files named DTLNG lng where lng is a three-character code that specifies the language. These files must be accessible for each language into which you want to translate dates.
- ❑ For these NLS characters to appear correctly, the Server and Client must be configured with the correct code pages.
- ❑ The DATETRAN function is not supported in Dialogue Manager.

Example: Using the DATETRAN Function

The following request prints the day of the week in the default case of the specific language:

```

DEFINE FILE VIDEOTRK
TRANS1/YYMD=20050104;
TRANS2/YYMD=20051003;

DATEW/W=TRANS1      ;
DATEW2/W=TRANS2     ;
DATEYYMD/YYMDW=TRANS1 ;
DATEYYMD2/YYMDW=TRANS2 ;

OUT1A/A8=DATETRAN(DATEW, '(W)', '(wr)', 'EN', 8, 'A8') ;
OUT1B/A8=DATETRAN(DATEW2, '(W)', '(wr)', 'EN', 8, 'A8') ;
OUT1C/A8=DATETRAN(DATEW, '(W)', '(wr)', 'ES', 8, 'A8') ;
OUT1D/A8=DATETRAN(DATEW2, '(W)', '(wr)', 'ES', 8, 'A8') ;
OUT1E/A8=DATETRAN(DATEW, '(W)', '(wr)', 'FR', 8, 'A8') ;
OUT1F/A8=DATETRAN(DATEW2, '(W)', '(wr)', 'FR', 8, 'A8') ;
OUT1G/A8=DATETRAN(DATEW, '(W)', '(wr)', 'DE', 8, 'A8') ;
OUT1H/A8=DATETRAN(DATEW2, '(W)', '(wr)', 'DE', 8, 'A8') ;
END

TABLE FILE VIDEOTRK
HEADING
"FORMAT wr"
" "
"Full day of week name at beginning of date, default case (wr)"
"English / Spanish / French / German"
" "
SUM OUT1A AS '' OUT1B AS '' TRANSDATE NOPRINT
OVER OUT1C AS '' OUT1D AS ''
OVER OUT1E AS '' OUT1F AS ''
OVER OUT1G AS '' OUT1H AS ''
ON TABLE SET PAGE-NUM OFF
ON TABLE SET STYLE *
GRID=OFF, $
END

```

The output is:

```
FORMAT wr
```

Full day of week name at beginning of date, default case (wr)
English / Spanish / French / German

Tuesday	Monday
martes	lunes
mardi	lundi
Dienstag	Montag

The following request prints a blank delimited date with an abbreviated month name in English. Initial zeros in the day number are suppressed, and a suffix is added to the end of the number:

```
DEFINE FILE VIDEOTRK
TRANS1/YYPD=20050104;
TRANS2/YYPD=20050302;

DATEW/W=TRANS1      ;
DATEW2/W=TRANS2     ;
DATEYYMD/YYPDW=TRANS1 ;
DATEYYMD2/YYPDW=TRANS2 ;

OUT2A/A15=DATETRAN(DATEYYMD, '(MDYY)', '(Btdo)', 'EN', 15, 'A15') ;
OUT2B/A15=DATETRAN(DATEYYMD2, '(MDYY)', '(Btdo)', 'EN', 15, 'A15') ;
END

TABLE FILE VIDEOTRK
HEADING
"FORMAT Btdo"
" "
"Blank-delimited (B)"
"Abbreviated month name, default case (t)"
"Zero-suppress day number, end with suffix (do)"
"English"
" "
SUM OUT2A AS '' OUT2B AS '' TRANSDATE NOPRINT
ON TABLE SET PAGE-NUM OFF
END
```

The output is:

FORMAT Btdo	
Blank-delimited (B)	
Abbreviated month name, default case (t)	
Zero-suppress day number, end with suffix (do)	
English	
Jan 4th 2005	Mar 2nd 2005

The following request prints a blank delimited date, with an abbreviated month name in German. Initial zeros in the day number are suppressed, and a period is added to the end of the number:

```

DEFINE FILE VIDEOTRK
TRANS1/YYMD=20050104;
TRANS2/YYMD=20050302;

DATEW/W=TRANS1      ;
DATEW2/W=TRANS2     ;
DATEYYMD/YYMDW=TRANS1  ;
DATEYYMD2/YYMDW=TRANS2 ;

OUT3A/A12=DATETRAN(DATEYYMD, '(DMYY)', '(Btdp)', 'DE', 12, 'A12');
OUT3B/A12=DATETRAN(DATEYYMD2, '(DMYY)', '(Btdp)', 'DE', 12, 'A12');
END

TABLE FILE VIDEOTRK
HEADING
"FORMAT Btdp"
""
"Blank-delimited (B)"
"Abbreviated month name, default case (t)"
"Zero-suppress day number, end with period (dp)"
"German"
""
SUM OUT3A AS '' OUT3B AS '' TRANSDATE NOPRINT
ON TABLE SET PAGE-NUM OFF
END

```

The output is:

FORMAT Btdp	
Blank-delimited (B)	
Abbreviated month name, default case (t)	
Zero-suppress day number, end with period (dp)	
German	
4. Jan 2005	2. Mär 2005

The following request prints a blank delimited date in French, with a full day name at the beginning and a full month name, in lowercase (the default for French):

```

DEFINE FILE VIDEOTRK
TRANS1/YYMD=20050104;
TRANS2/YYMD=20050302;

DATEW/W=TRANS1      ;
DATEW2/W=TRANS2     ;
DATEYYMD/YYMDW=TRANS1  ;
DATEYYMD2/YYMDW=TRANS2 ;

OUT4A/A30 = DATETRAN(DATEYYMD, '(DMYY)', '(Bwrtr)', 'FR', 30, 'A30');
OUT4B/A30 = DATETRAN(DATEYYMD2, '(DMYY)', '(Bwrtr)', 'FR', 30, 'A30');
END

TABLE FILE VIDEOTRK
HEADING
"FORMAT Bwrtr"
""
"Blank-delimited (B)"
"Full day of week name at beginning of date, default case (wr)"
"Full month name, default case (tr)"
"English"
""
SUM OUT4A AS '' OUT4B AS '' TRANSDATE NOPRINT
ON TABLE SET PAGE-NUM OFF
END

```


The output is:

FORMAT Bwrtr	
Blank-delimited (B)	
Full day of week name at beginning of date, default case (wr)	
Full month name, default case (tr)	
English	
mardi 04 janvier 2005	mercredi 02 mars 2005

The following request prints a blank delimited date in Spanish with a full day name at the beginning in lowercase (the default for Spanish), followed by a comma, and with the word “de” between the day number and month and between the month and year:

```

DEFINE FILE VIDEOTRK
TRANS1/YYMD=20050104;
TRANS2/YYMD=20050302;

DATEW/W=TRANS1      ;
DATEW2/W=TRANS2     ;
DATEYYMD/YYMDW=TRANS1  ;
DATEYYMD2/YYMDW=TRANS2 ;

OUT5A/A30=DATETRAN(DATEYYMD, '(DMYY)', '(Bwrctrde)', 'ES', 30, 'A30');
OUT5B/A30=DATETRAN(DATEYYMD2, '(DMYY)', '(Bwrctrde)', 'ES', 30, 'A30');
END

TABLE FILE VIDEOTRK
HEADING
"FORMAT Bwrctrde"
""
"Blank-delimited (B)"
"Full day of week name at beginning of date, default case (wr)"
"Comma after day name (c)"
"Full month name, default case (tr)"
"Zero-suppress day number (d)"
"de between day and month and between month and year (e)"
"Spanish"
""
SUM OUT5A AS '' OUT5B AS '' TRANSDATE NOPRINT
ON TABLE SET PAGE-NUM OFF
END

```

The output is:

FORMAT Bwrctrde	
Blank-delimited (B)	
Full day of week name at beginning of date, default case (wr)	
Comma after day name (c)	
Full month name, default case (tr)	
Zero-suppress day number (d)	
de between day and month and between month and year (e)	
Spanish	
martes, 4 de enero de 2005	miércoles, 2 de marzo de 2005

The following request prints a date in Japanese characters with a full month name at the beginning, in the default case and with zero suppression:

```

DEFINE FILE VIDEOTRK
TRANS1/YYMD=20050104;
TRANS2/YYMD=20050302;

DATEW/W=TRANS1      ;
DATEW2/W=TRANS2     ;
DATEYYMD/YYMDW=TRANS1 ;
DATEYYMD2/YYMDW=TRANS2 ;

OUT6A/A30=DATETRAN(DATEYYMD , '(YYMD)', '(Ktrd)', 'JA', 30, 'A30');
OUT6B/A30=DATETRAN(DATEYYMD2, '(YYMD)', '(Ktrd)', 'JA', 30, 'A30');
END

TABLE FILE VIDEOTRK
HEADING
"FORMAT Ktrd"
" "
"Japanese characters (K in conjunction with the language code JA)"
"Full month name at beginning of date, default case (tr)"
"Zero-suppress day number (d)"
"Japanese"
" "
SUM OUT6A AS ' ' OUT6B AS ' ' TRANSDATE NOPRINT
ON TABLE SET PAGE-NUM OFF
END

```

The output is:

FORMAT Ktrd Japanese characters (K in conjunction with the language code JA) Full month name at beginning of date, default case (tr) Zero-suppress day number (d) Japanese	
2005年1月4日	2005年3月2日

The following request prints a blank delimited date in Greek with a full day name at the beginning in the default case, followed by a comma, and with a full month name in the default case:

```

DEFINE FILE VIDEOTRK
TRANS1/YYMD=20050104;
TRANS2/YYMD=20050302;

DATEW/W=TRANS1      ;
DATEW2/W=TRANS2     ;
DATEYYMD/YYMDW=TRANS1 ;
DATEYYMD2/YYMDW=TRANS2 ;

OUT7A/A30=DATETRAN(DATEYYMD , '(DMYY)', '(Bwrctr)', 'GR', 30, 'A30');
OUT7B/A30=DATETRAN(DATEYYMD2, '(DMYY)', '(Bwrctr)', 'GR', 30, 'A30');
END

TABLE FILE VIDEOTRK
HEADING
"FORMAT Bwrctrde"
" "
"Blank-delimited (B)"
"Full day of week name at beginning of date, default case (wr)"
"Comma after day name (c)"
"Full month name, default case (tr)"
"Greek"
" "
SUM OUT7A AS ' ' OUT7B AS ' ' TRANSDATE NOPRINT
ON TABLE SET PAGE-NUM OFF
END

```

The output is:

FORMAT Bwrctr	
Blank-delimited (B)	
Full day of week name at beginning of date, default case (wr)	
Comma after day name (c)	
Full month name, default case (tr)	
Greek	
Τρίτη, 04 Ιανουάριος 2005	Τετάρτη, 02 Μάρτιος 2005

HDATE: Converting the Date Portion of a Date-Time Value to a Date Format

Available Languages: reporting, Maintain

The HDATE function converts the date portion of a date-time value to the date format YYMD. You can then convert the result to other date formats.

Syntax: How to Convert the Date Portion of a Date-Time Value to a Date Format

HDATE(datetime, output)

where:

datetime

Date-time

Is the date-time value to be converted, the name of a date-time field that contains the value, or an expression that returns the value.

output

Date

Is the format in single quotation marks or the field that contains the result.

Example: Converting the Date Portion of a Date-Time Field to a Date Format (Reporting)

HDATE converts the date portion of the TRANSDATE field to the date format YYMD:

```
TABLE FILE VIDEOTR2
PRINT CUSTID TRANSDATE AS 'DATE-TIME' AND COMPUTE
TRANSDATE_DATE/YYMD = HDATE(TRANSDATE, 'YYMD');
WHERE DATE EQ 2000;
END
```

The output is:

CUSTID	DATE-TIME	TRANSDATE_DATE
-----	-----	-----
1237	2000/02/05 03:30	2000/02/05
1118	2000/06/26 05:45	2000/06/26

Example: Converting the Date Portion of a Date-Time Field to a Date Format (Maintain)

HDATE converts the date portion of DT1 to date format YYMD:

```
MAINTAIN FILE DATETIME
FOR 1 NEXT ID INTO STK;
COMPUTE
DT1_DATE/YYMD = HDATE(STK.DT1, DT1_DATE);
TYPE "STK(1).DT1 = <STK(1).DT1";
TYPE "DT1_DATE = <DT1_DATE";
END
```

The output is:

```
STK(1).DT1 = 2000/1/1 02:57:25
DT1_DATE = 2000/01/01
```

HEXBYT: Converting a Decimal Integer to a Character

Available Languages: reporting, Maintain

The HEXBYT function obtains the ASCII, EBCDIC, or Unicode character equivalent of a decimal integer, depending on your configuration and operating environment. It returns a single alphanumeric character in the ASCII, EBCDIC, or Unicode character set. You can use this function to produce characters that are not on your keyboard, similar to the CTRAN function.

In Unicode configurations, this function uses values in the range:

- ☐ 0 to 255 for 1-byte characters.
- ☐ 256 to 65535 for 2-byte characters.
- ☐ 65536 to 16777215 for 3-byte characters.
- ☐ 16777216 to 4294967295 for 4-byte characters (primarily for EBCDIC).

The display of special characters depends on your software and hardware; not all special characters may appear.

Syntax: **How to Convert a Decimal Integer to a Character**

```
HEXBYT(decimal_value, output)
```

where:

decimal_value

Integer

Is the decimal integer to be converted to a single character. In non-Unicode environments, a value greater than 255 is treated as the remainder of *decimal_value* divided by 256.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks (').

Example: **Converting a Decimal Integer to a Character**

HEXBYT converts LAST_INIT_CODE to its character equivalent and stores the result in LAST_INIT:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND
COMPUTE LAST_INIT_CODE/I3 = BYTVAL(LAST_NAME, 'I3');
COMPUTE LAST_INIT/A1 = HEXBYT(LAST_INIT_CODE, LAST_INIT) ;
WHERE DEPARTMENT EQ 'MIS';
END
```

The output for an ASCII platform is:

LAST_NAME	LAST_INIT_CODE	LAST_INIT
SMITH	83	S
JONES	74	J
MCCOY	77	M
BLACKWOOD	66	B
GREENSPAN	71	G
CROSS	67	C

The output for an EBCDIC platform is:

LAST_NAME	LAST_INIT_CODE	LAST_INIT
SMITH	226	S
JONES	209	J
MCCOY	212	M
BLACKWOOD	194	B
GREENSPAN	199	G
CROSS	195	C

HEXTYPE: Returning the Hexadecimal View of an Input Value

The HEXTYPE function returns the hexadecimal view of an input value of any data type. The result is returned as variable length alphanumeric. The alphanumeric field to which the hexadecimal value is returned must be large enough to hold two characters for each input character. The value returned depends on the running operating environment.

Syntax: How to Returning the Hexadecimal View of an Input Value

```
HEXTYPE(in_value)
```

where:

in_value

Is an alphanumeric or integer field, constant, or expression.

Example: Returning a Hexadecimal View

The following request returns a hexadecimal view of the country names and the sum of the days delayed.

```
DEFINE FILE WF_RETAIL_LITE
Days/I8 = DAYSDELAYED;
Country/A20 = COUNTRY_NAME;
HexCountry/A30 = HEXTYPE(Country);
END
TABLE FILE WF_RETAIL_LITE
SUM COUNTRY_NAME NOPRINT Country HexCountry Days
COMPUTE HexDays/A40 = HEXTYPE(Days);
BY COUNTRY_NAME NOPRINT
WHERE COUNTRY_NAME LT 'P'
ON TABLE SET PAGE NOPAGE
END
```

The output is shown in the following image.

Country	HexCountry	Days	HexDays
Argentina	417267656E74696E6120202020202020	84	00000054
Australia	4175737472616C696120202020202020	27	0000001B
Austria	41757374726961202020202020202020	798	0000031E
Belgium	42656C6769756D202020202020202020	14	0000000E
Brazil	4272617A696C20202020202020202020	204	000000CC
Canada	43616E61646120202020202020202020	584	00000248
Chile	4368696C652020202020202020202020	45	0000002D
China	4368696E612020202020202020202020	1	00000001
Colombia	436F6C6F6D6269612020202020202020	114	00000072
Denmark	44656E6D61726B202020202020202020	0	00000000
Egypt	45677970742020202020202020202020	3	00000003
Finland	46696E6C616E64202020202020202020	3	00000003
France	4672616E636520202020202020202020	49	00000031
Germany	4765726D616E79202020202020202020	498	000001F2
Greece	47726565636520202020202020202020	9	00000009
Hungary	48756E67617279202020202020202020	7	00000007
India	496E6469612020202020202020202020	23	00000017
Ireland	4972656C616E64202020202020202020	7	00000007
Israel	49737261656C20202020202020202020	2	00000002
Italy	4974616C792020202020202020202020	7	00000007
Japan	4A6170616E2020202020202020202020	12	0000000C
Luxembourg	4C7578656D626F757267202020202020	0	00000000
Malaysia	4D616C61797369612020202020202020	20	00000014
Mexico	4D657869636F20202020202020202020	170	000000AA
Netherlands	4E65746865726C616E64732020202020	8	00000008
Norway	4E6F7277617920202020202020202020	0	00000000

JPTRANS: Converting Japanese Specific Characters

The JPTRANS function converts Japanese specific characters.

Syntax: **How to Convert Japanese Specific Characters**

```
JPTRANS ('type_of_conversion', length, source_string, 'output_format')
```

where:

type_of_conversion

Is one of the following options indicating the type of conversion you want to apply to Japanese specific characters. The following table shows the single component input types:

Conversion Type	Description
'UPCASE'	Converts Zenkaku (Fullwidth) alphabets to Zenkaku uppercase.
'LOCASE'	Converts Zenkaku alphabets to Zenkaku lowercase.
'HNZNALPHA'	Converts alphanumerics from Hankaku (Halfwidth) to Zenkaku.
'HNZNSIGN'	Converts ASCII symbols from Hankaku to Zenkaku.
'HNZNKANA'	Converts Katakana from Hankaku to Zenkaku.
'HNZNSPACE'	Converts space (blank) from Hankaku to Zenkaku.
'ZNHNALPHA'	Converts alphanumerics from Zenkaku to Hankaku.
'ZNHNSIGN'	Converts ASCII symbols from Zenkaku to Hankaku.
'ZNHNKANA'	Converts Katakana from Zenkaku to Hankaku.
'ZNHNSPACE'	Converts space from Zenkaku to Hankaku.
'HIRAKATA'	Converts Hiragana to Zenkaku Katakana.
'KATAHIRA'	Converts Zenkaku Katakana to Hiragana.
'930T0939'	Converts codepage from 930 to 939.
'939T0930'	Converts codepage from 939 to 930.

length

Integer

Is the number of characters in the source_string.

source_string

Alphanumeric

Is the string to convert.

output_format

Alphanumeric

Is the name of the field that contains the output, or the format enclosed in single quotation marks (').

Example: Using the JPTRANS Function

```
JPTRANS('UPCASE', 20, Alpha_DBCS_Field, 'A20')
```

For a b c , the result is A B C .

```
JPTRANS('LOCASE', 20, Alpha_DBCS_Field, 'A20')
```

For A B C , the result is a b c .

```
JPTRANS('HNZNALPHA', 20, Alpha_SBCS_Field, 'A20')
```

For AaBbCc123, the result is A a B b C c 1 2 3 .

```
JPTRANS('HNZNSIGN', 20, Symbol_SBCS_Field, 'A20')
```

For !@\$%&.,?, the result is ! @ \$ % 、 。 ?

```
JPTRANS('HNZNKANA', 20, Hankaku_Katakana_Field, 'A20')
```

For 「^ -あ* -ル。 」, the result is 「ベースボール。」

```
JPTRANS('HNZNSPACE', 20, Hankaku_Katakana_Field, 'A20')
```

For アイウ, the result is ア イ ウ

```
JPTRANS('ZNHNALPHA', 20, Alpha_DBCS_Field, 'A20')
```

For A a B b C c 1 2 3 , the result is AaBbCc123.

```
JPTRANS('ZNHNSIGN', 20, Symbol_DBCS_Field, 'A20')
```

For ! @ \$ % \ , . ? , the result is ! @ \$ % , . ?

```
JPTRANS('ZNHNKANA', 20, Zenkaku_Katakana_Field, 'A20')
```

For 「ベースボール。」 , the result is 「^ - 入 本 - ル。」

```
JPTRANS('ZNHNSPACE', 20, Zenkaku_Katakana_Field, 'A20')
```

For ア イ ウ , the result is アイウ

```
JPTRANS('HIRAKATA', 20, Hiragana_Field, 'A20')
```

For あいう , the result is アイウ

```
JPTRANS('KATAHIRA', 20, Zenkaku_Katakana_Field, 'A20')
```

For アイウ , the result is あいう

In the following, codepoints 0x62 0x63 0x64 are converted to 0x81 0x82 0x83, respectively:

```
JPTRANS('930TO939', 20, CP930_Field, 'A20')
```

In the following, codepoints 0x59 0x62 0x63 are converted to 0x81 0x82 0x83, respectively:

```
JPTRANS('939TO930', 20, CP939_Field, 'A20')
```

Reference: Usage Notes for the JPTRANS Function

- ❑ HNZNSIGN and ZNHNSIGN focus on the conversion of symbols.

Many symbols have a one-to-one relation between Japanese Fullwidth characters and ASCII symbols, whereas some characters have one-to-many relations. For example, the Japanese punctuation character (U+3001) and Fullwidth comma , (U+FF0C) will be converted to the same comma , (U+002C). The following EXTRA rule for those special cases is shown below:

HNZNSIGN:

- ❑ Double Quote " (U+0022) -> Fullwidth Right Double Quote ” (U+201D)
- ❑ Single Quote ' (U+0027) -> Fullwidth Right Single Quote ’ (U+2019)
- ❑ Comma , (U+002C) -> Fullwidth Ideographic Comma (U+3001)

- ☐ Full Stop . (U+002E) -> Fullwidth Ideographic Full Stop ? (U+3002)
- ☐ Backslash \ (U+005C) -> Fullwidth Backslash \ (U+FF3C)
- ☐ Halfwidth Left Corner Bracket (U+FF62) -> Fullwidth Left Corner Bracket (U+300C)
- ☐ Halfwidth Right Corner Bracket (U+FF63) -> Fullwidth Right Corner Bracket (U+300D)
- ☐ Halfwidth Katakana Middle Dot ? (U+FF65) -> Fullwidth Middle Dot · (U+30FB)

ZNHNSIGN:

- ☐ Fullwidth Right Double Quote ” (U+201D) -> Double Quote " (U+0022)
 - ☐ Fullwidth Left Double Quote “ (U+201C) -> Double Quote " (U+0022)
 - ☐ Fullwidth Quotation " (U+FF02) -> Double Quote " (U+0022)
 - ☐ Fullwidth Right Single Quote ’ (U+2019) -> Single Quote ' (U+0027)
 - ☐ Fullwidth Left Single Quote ‘ (U+2018) -> Single Quote ' (U+0027)
 - ☐ Fullwidth Single Quote ' (U+FF07) -> Single Quote ' (U+0027)
 - ☐ Fullwidth Ideographic Comma (U+3001) -> Comma , (U+002C)
 - ☐ Fullwidth Comma , (U+FF0C) -> Comma , (U+002C)
 - ☐ Fullwidth Ideographic Full Stop ? (U+3002) -> Full Stop . (U+002E)
 - ☐ Fullwidth Full Stop . (U+FF0E) -> Full Stop . (U+002E)
 - ☐ Fullwidth Yen Sign ¥ (U+FFE5) -> Yen Sign ¥ (U+00A5)
 - ☐ Fullwidth Backslash \ (U+FF3C) -> Backslash \ (U+005C)
 - ☐ Fullwidth Left Corner Bracket (U+300C) -> Halfwidth Left Corner Bracket (U+FF62)
 - ☐ Fullwidth Right Corner Bracket (U+300D) -> Halfwidth Right Corner Bracket (U+FF63)
 - ☐ Fullwidth Middle Dot · (U+30FB) -> Halfwidth Katakana Middle Dot · (U+FF65)
- ☐ HNZNKANA and ZHNKANA focus on the conversion of Katakana
- They convert not only letters, but also punctuation symbols on the following list:
- ☐ Fullwidth Ideographic Comma (U+3001) <-> Halfwidth Ideographic Comma (U+FF64)

- ❑ Fullwidth Ideographic Full Stop (U+3002) <-> Halfwidth Ideographic Full Stop (U+FF61)
- ❑ Fullwidth Left Corner Bracket (U+300C) <-> Halfwidth Left Corner Bracket (U+FF62)
- ❑ Fullwidth Right Corner Bracket (U+300D) <-> Halfwidth Right Corner Bracket (U+FF63)
- ❑ Fullwidth Middle Dot · (U+30FB) <-> Halfwidth Katakana Middle Dot · (U+FF65)
- ❑ Fullwidth Prolonged Sound (U+30FC) <-> Halfwidth Prolonged Sound (U+FF70)
- ❑ JPTRANS can be nested for multiple conversions.

For example, text data may contain fullwidth numbers and fullwidth symbols. In some situations, they should be cleaned up for ASCII numbers and symbols.

For バンゴウ# 1 2 3 , the result is バンゴウ#123

```
JPTRANS('ZHNHALPHA', 20, JPTRANS('ZHNNSIGN', 20, Symbol_DBCS_Field,
'A20'), 'A20')
```

- ❑ HNZNSPACE and ZHNNSPACE focus on the conversion of a space (blank character).

Currently only conversion between U+0020 and U+3000 is supported.

KKFCUT: Truncating a String

If your configuration uses a DBCS code page, you can use the KKFCUT function to truncate a string.

Syntax: How to Truncate a String

```
KKFCUT(length, source_string, output)
```

where:

length

Integer

Is the length of the source string in *bytes*, or a field that contains the length. The string can have a mixture of DBCS and SBCS characters. Therefore, the number of bytes represents the maximum number of characters possible in the source string.

source_string

Alphanumeric

Is the string that will be truncated enclosed in single quotation marks ('), or the field containing the string.

output

Alphanumeric

Is the field to which the result is returned, or the format of the output value enclosed in single quotation marks (').

The string will be truncated to the number of bytes in the output field.

Example: Truncating a String

In the following, KKFCUT truncates the COUNTRY field (up to 10 bytes long) to A4 format:

```
COUNTRY_CUT/A4 = KKFCUT(10, COUNTRY, 'A4');
```

The output in ASCII environments is shown in the following image:

国名	COUNTRY_CUT
-----	-----
イギリス	イギ
日本	日本
イタリア	イタ
ドイツ	ドイ
フランス	フラ

The output in EBCDIC environments is shown in the following image:

国名	COUNTRY_CUT
-----	-----
イギリス	イ
日本	日
イタリア	イ
ドイツ	ド
フランス	フ

LCWORD: Converting a String to Mixed-Case

Available Languages: reporting, Maintain

The LCWORD function converts the letters in a character string to mixed-case. It converts every alphanumeric character to lowercase except the first letter of each new word and the first letter after a single or double quotation mark, which it converts to uppercase. For example, O'CONNOR is converted to O'Connor and JACK'S to Jack'S.

LCWORD skips numeric and special characters in the source string and continues to convert the following alphabetic characters. The result of LCWORD is a string in which the initial uppercase characters of all words are followed by lowercase characters.

Syntax: **How to Convert a Character String to Mixed-Case**

```
LCWORD(length, source_string, output)
```

where:

length

Integer

Is the number of characters in *source_string* and *output*.

string

Alphanumeric

Is the character string to be converted enclosed in single quotation marks, or a field or variable containing the character string.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks. The length must be greater than or equal to *length*.

Example: **Converting a Character String to Mixed-Case**

LCWORD converts the LAST_NAME field to mixed-case and stores the result in MIXED_CASE.

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND COMPUTE
MIXED_CASE/A15 = LCWORD(15, LAST_NAME, MIXED_CASE) ;
WHERE DEPARTMENT EQ 'PRODUCTION'
END
```

The output is:

LAST_NAME	MIXED_CASE
-----	-----
STEVENS	Stevens
SMITH	Smith
BANNING	Banning
IRVING	Irving
ROMANS	Romans
MCKNIGHT	Mcknight

LCWORD2: Converting a String to Mixed-Case

Available Languages: reporting, Maintain

The LCWORD2 function converts the letters in a character string to mixed-case by converting the first letter of each word to uppercase and converting every other letter to lowercase. In addition, a double quotation mark or a space indicates that the next letter should be converted to uppercase.

For example, "SMITH" would be changed to "Smith" and "JACK S" would be changed to "Jack S".

Syntax: How to Convert a Character String to Mixed-Case

`LCWORD2(length, string, output)`

where:

length

Integer

Is the length, in characters, of the character string or field to be converted, or a field that contains the length.

string

Alphanumeric

Is the character string to be converted, or a temporary field that contains the string.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks. The length must be greater than or equal to *length*.

Example: Converting a Character String to Mixed-Case

LCWORD2 converts the string O'CONNOR's to mixed-case:

```
DEFINE FILE EMPLOYEE
MYVAL1/A10='O'CONNOR'S';
LC2/A10 = LCWORD2(10, MYVAL1, 'A10');
END
TABLE FILE EMPLOYEE
SUM LAST_NAME NOPRINT MYVAL1 LC2
END
```


The output is:

```
MYVAL1      LC2
-----
O 'CONNOR' S  O 'Connor' s
```

LCWORD3: Converting a String to Mixed-Case

The LCWORD3 function converts the letters in a character string to mixed-case by converting the first letter of each word to uppercase and converting every other letter to lowercase. In addition, a single quotation mark indicates that the next letter should be converted to uppercase, as long as it is neither followed by a blank nor the last character in the input string.

For example, 'SMITH' would be changed to 'Smith' and JACK'S would be changed to Jack's.

Syntax: How to Convert a Character String to Mixed-Case Using LCWORD3

```
LCWORD3(length, string, output)
```

where:

length

Integer

Is the length, in characters, of the character string or field to be converted, or a field that contains the length.

string

Alphanumeric

Is the character string to be converted, or a field that contains the string.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks. The length must be greater than or equal to *length*.

Example: Converting a Character String to Mixed-Case Using LCWORD3

LCWORD3 converts the strings O'CONNOR's and o'connor's to mixed-case:

```
DEFINE FILE EMPLOYEE
MYVAL1/A10='O'CONNOR'S';
MYVAL2/A10='o'connor's';
LC1/A10 = LCWORD3(10, MYVAL1, 'A10');
LC2/A10 = LCWORD3(10, MYVAL2, 'A10');
END
TABLE FILE EMPLOYEE
SUM LAST_NAME NOPRINT MYVAL1 LC1 MYVAL2 LC2
END
```

On the output, the letter C after the first single quotation mark is in uppercase because it is not followed by a blank and is not the final letter in the input string. The letter s after the second single quotation mark (') is in lowercase because it is the last character in the input string:

MYVAL1	LC1	MYVAL2	LC2
-----	---	-----	---
O'CONNOR'S	O'Connor's	o'connor's	O'Connor's

LOCASE: Converting Text to Lowercase

Available Languages: reporting, Maintain

The LOCASE function converts alphanumeric text to lowercase.

Syntax: How to Convert Text to Lowercase

LOCASE(length, source_string, output)

where:

length

Integer

Is the number of characters in *source_string* and *output*, or a field that contains the length. The length must be greater than 0 and the same for both arguments; otherwise, an error occurs.

source_string

Alphanumeric

Is the character string to convert in single quotation marks, or a field or variable that contains the string.

output

Alphanumeric

Is the name of the field in which to store the result, or the format of the output value enclosed in single quotation marks. The field name can be the same as *source_string*.

Example: Converting a String to Lowercase

LOCASE converts the LAST_NAME field to lowercase and stores the result in LOWER_NAME:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME AND COMPUTE
LOWER_NAME/A15 = LOCASE(15, LAST_NAME, LOWER_NAME);
WHERE DEPARTMENT EQ 'MIS';
END
```

The output is:

LAST_NAME	LOWER_NAME
-----	-----
SMITH	smith
JONES	jones
MCCOY	mccoy
BLACKWOOD	blackwood
GREENSPAN	greenspan
CROSS	cross

PATTERN: Generating a Pattern From a String

The PATTERN function examines a source string and produces a pattern that indicates the sequence of numbers, uppercase letters, and lowercase letters in the source string. This function is useful for examining data to make sure that it follows a standard pattern.

In the output pattern:

- ❑ Any character from the input that represents a single-byte digit becomes the character 9.
- ❑ Any character that represents an uppercase letter becomes A, and any character that represents a lowercase letter becomes a. For European NLS mode (Western Europe, Central Europe), A and a are extended to apply to accented alphabets.
- ❑ For Japanese, double-byte characters and Hankaku-katakana become C (uppercase). Note that double-byte includes Hiragana, Katakana, Kanji, full-width alphabets, full-width numbers, and full-width symbols. This means that all double-byte letters such as Chinese and Korean are also represented as C.
- ❑ Special characters remain unchanged.
- ❑ An unprintable character becomes the character X.

Syntax: How to Generate a Pattern From an Input String

```
PATTERN (length, source_string, output)
```

where:

length

Numeric

Is the length of *source_string*.

source_string

Alphanumeric

Is the source string enclosed in single quotation marks, or a field containing the source string.

output

Alphanumeric

Is the name of the field to contain the result or the format of the field enclosed in single quotation marks.

Example: Producing a Pattern From Alphanumeric Data

The following 19 records are stored in a fixed format sequential file (with LRECL 14) named TESTFILE:

```
212-736-6250
212 736 4433
123-45-6789
800-969-INFO
10121-2898
10121
2 Penn Plaza
917-339-6380
917-339-4350
(212) 736-6250
(212) 736-4433
212-736-6250
212-736-6250
212-736-6250
(212) 736 5533
(212) 736 5533
(212) 736 5533
10121 Æ
800-969-INFO
```

The Master File is:

```
FILENAME=TESTFILE, SUFFIX=FIX ,
  SEGMENT=TESTFILE, SEGTYPE=S0, $
    FIELDNAME=TESTFLD, USAGE=A14, ACTUAL=A14, $
```

The following request generates a pattern for each instance of TESTFLD and displays them by the pattern that was generated. It shows the count of each pattern and its percentage of the total count. The PRINT command shows which values of TESTFLD generated each pattern.

```
FILEDEF TESTFILE DISK testfile.ftmDEFINE FILE TESTFILE
  PATTERN/A14 = PATTERN (14, TESTFLD, 'A14' ) ;
END
TABLE FILE TESTFILE
  SUM CNT.PATTERN AS 'COUNT' PCT.CNT.PATTERN AS 'PERCENT'
  BY PATTERN
  PRINT TESTFLD
  BY PATTERN
ON TABLE COLUMN-TOTAL
END
```

Note that the next to last line produced a pattern from an input string that contained an unprintable character, so that character was changed to X. Otherwise, each numeric digit generated a 9 in the output string, each uppercase letter generated the character 'A', and each lowercase letter generated the character 'a'. The output is:

PATTERN	COUNT	PERCENT	TESTFLD
-----	-----	-----	-----
(999) 999 9999	3	15.79	(212) 736 5533 (212) 736 5533 (212) 736 5533
(999) 999-9999	2	10.53	(212) 736-6250 (212) 736-4433
9 Aaaa Aaaaa	1	5.26	2 Penn Plaza
999 999 9999	1	5.26	212 736 4433
999-99-9999	1	5.26	123-45-6789
999-999-AAAA	2	10.53	800-969-INFO 800-969-INFO
999-999-9999	6	31.58	212-736-6250 917-339-6380 917-339-4350 212-736-6250 212-736-6250 212-736-6250
99999	1	5.26	10121
99999 X	1	5.26	10121 ¤
99999-9999	1	5.26	10121-2898
TOTAL	19	100.00	

REPLACE: Replacing a String

REPLACE replaces all instances of a search string in an input string with the given replacement string. The output is always variable length alphanumeric with a length determined by the input parameters.

Syntax: How to Replace all Instances of a String

```
REPLACE(input_string , search_string , replacement)
```

where:

input_string

Alphanumeric or text (An, AnV, TX)

Is the input string.

search_string

Alphanumeric or text (An, AnV, TX)

Is the string to search for within the input string.

replacement

Alphanumeric or text (An, AnV, TX)

Is the replacement string to be substituted for the search string. It can be a null string ('').

Example: Replacing a String

REPLACE replaces the string 'South' in the Country Name with the string 'S.'

```
SET TRACEUSER = ON
SET TRACEON = STMTRACE//CLIENT
SET TRACESTAMP=OFF
DEFINE FILE WF_RETAIL_LITE
NEWNAME/A20 = REPLACE(COUNTRY_NAME, 'SOUTH', 'S. ');
END
TABLE FILE WF_RETAIL_LITE
SUM COUNTRY_NAME
BY NEWNAME AS 'New,Name'
WHERE COUNTRY_NAME LIKE 'S%'
ON TABLE SET PAGE NOLEAD
END
```

The output is shown in the following image.

New Name	Customer Country
S. Africa	South Africa
S. Korea	South Korea
Singapore	Singapore
Spain	Spain
Sweden	Sweden
Switzerland	Switzerland

Example: Replacing All Instances of a String

In the following request, the virtual field DAYNAME1 is the string DAY1 with all instances of the string 'DAY' replaced with the string 'day'. The virtual field DAYNAME2 has all instances of the string 'DAY' removed.

```
DEFINE FILE WF_RETAIL
DAY1/A30 = 'SUNDAY MONDAY TUESDAY';
DAYNAME1/A30 = REPLACE(DAY1, 'DAY', 'day' );
DAYNAME2/A30 = REPLACE(DAY1, 'DAY', ' ');
END
TABLE FILE WF_RETAIL
PRINT DAY1 OVER
DAYNAME1 OVER
DAYNAME2
WHERE EMPLOYEE_NUMBER EQ 'AH118'
ON TABLE SET PAGE NOPAGE
END
```

The output is:

```
DAY1          SUNDAY MONDAY TUESDAY
DAYNAME1      SUNday MONday TUESday
DAYNAME2      SUN MON TUES
```

REVERSE: Reversing the Characters in a String

The REVERSE function reverses the characters in a string. This reversal includes all trailing blanks, which then become leading blanks. However, in an HTML report with SET SHOWBLANKS=OFF (the default value), the leading blanks are not visible.

Syntax: How to Reverse the Characters in a String

```
REVERSE(length, source_string, output)
```

where:

length

Integer

Is the number of characters in *source_string* and *output*, or a field that contains the length.

source_string

Alphanumeric

Is the character string to reverse enclosed in single quotation marks, or a field that contains the character string.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks.

Example: Reversing the Characters in a String

In the following request against the EMPLOYEE data source, the REVERSE function is used to reverse the characters in the LAST_NAME field to produce the field named REVERSE_LAST. In this field, the trailing blanks from LAST_NAME have become leading blanks. The TRIM function is used to strip the leading blanks from REVERSE_LAST to produce the field named TRIM_REVERSE:

```
DEFINE FILE EMPLOYEE
REVERSE_LAST/A15 = REVERSE(15, LAST_NAME, REVERSE_LAST);
TRIM_REVERSE/A15 = TRIM('L', REVERSE_LAST, 15, ' ', 1, 'A15');
END
TABLE FILE EMPLOYEE
PRINT REVERSE_LAST TRIM_REVERSE
BY LAST_NAME
END
```


The output is:

LAST_NAME	REVERSE_LAST	TRIM_REVERSE
-----	-----	-----
BANNING	GNINNAB	GNINNAB
BLACKWOOD	DOOWKCALB	DOOWKCALB
CROSS	SSORC	SSORC
GREENSPAN	NAPSNEERG	NAPSNEERG
IRVING	GNIVRI	GNIVRI
JONES	SENOJ	SENOJ
MCCOY	YOCCM	YOCCM
MCKNIGHT	THGINKCM	THGINKCM
ROMANS	SNAMOR	SNAMOR
SMITH	HTIMS	HTIMS
	HTIMS	HTIMS
STEVENS	SNEVETS	SNEVETS

STRIP: Removing a Character From a String

Available Languages: reporting, Maintain

The STRIP function removes all occurrences of a specific character from a string. The resulting character string has the same length as the original string but is padded on the right with spaces.

Syntax: How to Remove a Character From a String

STRIP(length, source_string, char, output)

where:

length

Integer

Is the number of characters in *source_string* and *output*, or a field that contains the number.

source_string

Alphanumeric

Is the string from which the character will be removed, or a field containing the string.

char

Alphanumeric

Is the character to be removed from the string. This can be an alphanumeric literal enclosed in single quotation marks, or a field that contains the character. If more than one character is provided, the left-most character will be used as the strip character.

Note: To remove single quotation marks, use two consecutive quotation marks. You must then enclose this character combination in single quotation marks.

output

Alphanumeric

Is the field that contains the result, or the format of the output value enclosed in single quotation marks.

Example: Removing Occurrences of a Character From a String

STRIP removes all occurrences of a period (.) from the DIRECTOR field and stores the result in a field with the format A17:

```
TABLE FILE MOVIES
PRINT DIRECTOR AND COMPUTE
SDIR/A17 = STRIP(17, DIRECTOR, '.', 'A17');
WHERE CATEGORY EQ 'COMEDY'
END
```

The output is:

DIRECTORS	SDIR
-----	----
ZEMECKIS R.	ZEMECKIS R
ABRAHAMS J.	ABRAHAMS J
ALLEN W.	ALLEN W
HALLSTROM L.	HALLSTROM L
MARSHALL P.	MARSHALL P
BROOKS J.L.	BROOKS JL

Example: Removing Single Quotation Marks From a String

STRIP removes all occurrences of a single quotation mark (') from the TITLE field and stores the result in a field with the format A39:

```
TABLE FILE MOVIES
PRINT TITLE AND COMPUTE
STITLE/A39 = STRIP(39, TITLE, "'", 'A39');
WHERE TITLE CONTAINS "'"
END
```

The output is:

TITLE	STITLE
-----	-----
BABETTE'S FEAST	BABETTES FEAST
JANE FONDA'S COMPLETE WORKOUT	JANE FONDAS COMPLETE WORKOUT
JANE FONDA'S NEW WORKOUT	JANE FONDAS NEW WORKOUT
MICKEY MANTLE'S BASEBALLTIPS	MICKEY MANTLES BASEBALL TIPS

Example: Removing Commas From a String (Maintain)

STRIP removes all occurrences of a comma from the TITLE field:

```
MAINTAIN FILE MOVIES
FOR 10 NEXT MOVIECODE INTO MOVSTK
  WHERE TITLE CONTAINS ',';
COMPUTE I/I2=1;
REPEAT MOVSTK.FOCINDEX
TYPE "TITLE IS: <MOVSTK(I).TITLE"
COMPUTE NOCOMMA/A39=STRIP(39,MOVSTK().TITLE, ',',NOCOMMA);
TYPE "NEW TITLE IS: <NOCOMMA";
COMPUTE I=I+1
ENDREPEAT
END
```

The output is:

```
TITLE IS: SMURFS, THE
NEW TITLE IS: SMURFS THE
```

DSTRIP: Removing a Single-Byte or Double-Byte Character From a String

The DSTRIP function removes all occurrences of a specific single-byte or double-byte character from a string. The resulting character string has the same length as the original string, but is padded on the right with spaces.

Syntax: How to Remove a Single-Byte or Double-Byte Character From a String

```
DSTRIP(length, source_string, char, output)
```

where:

length

Double

Is the number of characters in *source_string* and *outfield*.

source_string

Alphanumeric

Is the string from which the character will be removed.

char

Alphanumeric

Is the character to be removed from the string. If more than one character is provided, the left-most character will be used as the strip character.

Note: To remove single quotation marks, use two consecutive quotation marks. You must then enclose this character combination in single quotation marks.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks (').

Example: Removing a Double-Byte Character From a String

In the following:

```
DSTRIP(9, 'A日A本B語', '日', A9)
```

For A日A本B語, the result is AA本B語.

SFTDEL: Deleting the Shift Code From DBCS Data

If your configuration uses a DBCS code page, you can use the SFTDEL function to delete the shift code from DBCS data.

Syntax: How to Delete the Shift Code From DBCS Data

```
SFTDEL(source_string, length, output)
```

where:

source_string

Alphanumeric

Is the string from which the shift code will be deleted enclosed in single quotation marks ('), or the field containing the string.

length

Integer

Is the length of the source string in *bytes*, or a field that contains the length. The string can have a mixture of DBCS and SBCS characters. Therefore, the number of bytes represents the maximum number of characters possible in the source string.

output

Alphanumeric

Is the field to which the result is returned, or the format of the output value enclosed in single quotation marks ('').

Example: Deleting the Shift Code From a String

In the following, SFTDEL deleted the shift code from the COUNTRY field (up to 10 bytes long):

```
COUNTRY_DEL/A10 = SFTDEL(COUNTRY, 10, 'A10');
```

The output in ASCII environments is shown in the following image:

国名	COUNTRY_DEL
-----	-----
イギリス	イギリス
日本	日本
イタリア	イタリア
ドイツ	ドイツ
フランス	フランス

The output in EBCDIC environments is shown in the following image:

国名	COUNTRY_DEL
-----	-----
イギリス	「b「A「メ「ヌ
日本	、イ、カ
イタリア	「b「j「メ「a
ドイツ	「「b「l
フランス	「ホ「「「「ヌ

SFTINS: Inserting the Shift Code Into DBCS Data

If your configuration uses a DBCS code page, you can use the SFTINS function to insert the shift code into DBCS data.

Syntax: **How to Insert the Shift Code Into DBCS Data**

```
SFTINS(source_string, length, output)
```

where:

```
source_string
```

Alphanumeric

Is the string into which the shift code will be inserted enclosed in single quotation marks ('), or the field containing the string.

```
length
```

Integer

Is the length of the source string in *bytes*, or a field that contains the length. The string can have a mixture of DBCS and SBCS characters. Therefore, the number of bytes represents the maximum number of characters possible in the source string.

```
output
```

Alphanumeric

Is the field to which the result is returned, or the format of the output value enclosed in single quotation marks (').

Example: **SFTINS: Inserting the Shift Code Into a String**

In the following example, SFTINS inserts the shift code into the COUNTRY_DEL field (which is the COUNTRY field with the shift code deleted):

```
COUNTRY_INS/A10 = SFTINS(COUNTRY_DEL, 10, 'A10');
```

The output displays the original COUNTRY field, the COUNTRY_DEL field with the shift code deleted, and the COUNTRY_INS field with the shift code re-inserted.

The output in ASCII environments, is shown in the following image:

国名	COUNTRY_DEL	COUNTRY_INS
-----	-----	-----
イギリス	イギリス	イギリス
日本	日本	日本
イタリア	イタリア	イタリア
ドイツ	ドイツ	ドイツ
フランス	フランス	フランス

The output in EBCDIC environments is shown in the following image:

国名	COUNTRY_DEL	COUNTRY_INS
イギリス	「b「A「メ「ヌ	イギリス
日本	、イ、カ	日本
イタリア	「b「j「メ「a	イタリア
ドイツ	「「b「l	ドイツ
フランス	「ホ「「「「ヌ	フランス

STRREP: Replacing Character Strings

The STRREP replaces all instances of a specified string within a source string. It also supports replacement by null strings.

Syntax: How to Replace Character Strings

STRREP (*inlength*, *instring*, *searchlength*, *searchstring*, *replength*, *repstring*, *outlength*, *output*)

where:

inlength

Numeric

Is the number of characters in the source string.

instring

Alphanumeric

Is the source string.

searchlength

Numeric

Is the number of characters in the (shorter length) string to be replaced.

searchstring

Alphanumeric

Is the character string to be replaced.

replength

Numeric

Is the number of characters in the replacement string. Must be zero (0) or greater.

repstring

Alphanumeric

Is the replacement string (alphanumeric). Ignored if replength is zero (0).

outlength

Numeric

Is the number of characters in the resulting output string. Must be 1 or greater.

output

Alphanumeric

Is the resulting output string after all replacements and padding.

Reference: Usage Note for STRREP Function

The maximum string length is 4095.

Example: Replacing Commas and Dollar Signs

In the following example, STRREP finds and replaces commas and dollar signs that appear in the CS_ALPHA field, first replacing commas with null strings to produce CS_NOCOMMAS (removing the commas) and then replacing the dollar signs (\$) with (USD) in the right-most CURR_SAL column:

```
TABLE FILE EMPLOYEE
SUM CURR_SAL NOPRINT
COMPUTE CS_ALPHA/A15=FTOA(CURR_SAL,'(D12.2M)',CS_ALPHA);
        CS_NOCOMMAS/A14=STRREP(15,CS_ALPHA,1,',',' ',0,'X',14,CS_NOCOMMAS);
        CS_USD/A17=STRREP(14,CS_NOCOMMAS,1,'$',4,'USD ',17,CS_USD);
        NOPRINT
        CS_USD/R AS CURR_SAL
BY LAST_NAME
END
```

The output is:

LAST_NAME	CS_ALPHA	CS_NOCOMMAS	CURR_SAL
BANNING	\$29,700.00	\$29700.00	USD 29700.00
BLACKWOOD	\$21,780.00	\$21780.00	USD 21780.00
CROSS	\$27,062.00	\$27062.00	USD 27062.00
GREENSPAN	\$9,000.00	\$9000.00	USD 9000.00
IRVING	\$26,862.00	\$26862.00	USD 26862.00
JONES	\$18,480.00	\$18480.00	USD 18480.00
MCCOY	\$18,480.00	\$18480.00	USD 18480.00
MCKNIGHT	\$16,100.00	\$16100.00	USD 16100.00
ROMANS	\$21,120.00	\$21120.00	USD 21120.00
SMITH	\$22,700.00	\$22700.00	USD 22700.00
STEVENS	\$11,000.00	\$11000.00	USD 11000.00

UFMT: Converting an Alphanumeric String to Hexadecimal

Available Languages: reporting, Maintain

The UFMT function converts characters in an alphanumeric source string to their hexadecimal representation. This function is useful for examining data of unknown format. As long as you know the length of the data, you can examine its content.

Syntax: How to Convert an Alphanumeric String to Hexadecimal

```
UFMT(source_string, length, output)
```

where:

source_string

Alphanumeric

Is the alphanumeric string to convert enclosed in single quotation marks ('), or the field that contains the string.

length

Integer

Is the number of characters in *source_string*.

output

Alphanumeric

Is the name of the field that contains the result, or the format of the output value enclosed in single quotation marks ('). The format of *output* must be alphanumeric and its length must be twice that of *length*.

Example: Converting an Alphanumeric String to Hexadecimal

UFMT converts each value in JOBCODE to its hexadecimal representation and stores the result in HEXCODE:

```
DEFINE FILE JOBCODE
HEXCODE/A6 = UFMT(JOBCODE, 3, HEXCODE);
END
TABLE FILE JOBCODE
PRINT JOBCODE HEXCODE
END
```

The output is:

JOBCODE	HEXCODE
-----	-----
A01	C1F0F1
A02	C1F0F2
A07	C1F0F7
A12	C1F1F2
A14	C1F1F4
A15	C1F1F5
A16	C1F1F6
A17	C1F1F7
B01	C2F0F1
B02	C2F0F2
B03	C2F0F3
B04	C2F0F4
B14	C2F1F4

UPCASE: Converting Text to Uppercase

Available Languages: reporting

The UPCASE function converts a character string to uppercase. It is useful for sorting on a field that contains both mixed-case and uppercase values. Sorting on a mixed-case field produces incorrect results because the sorting sequence in EBCDIC always places lowercase letters before uppercase letters, while the ASCII sorting sequence always places uppercase letters before lowercase. To obtain correct results, define a new field with all of the values in uppercase, and sort on that field.

***Syntax:* How to Convert Text to Uppercase**

UPCASE(length, source_string, output)

where:

length

Integer

Is the number of characters in *source_string* and *output*.

input

Alphanumeric

Is the string to convert enclosed in single quotation marks, or the field containing the character string.

output

Alphanumeric of type AnV or An

Is the field to which the result is returned, or the format of the output value enclosed in single quotation marks.

Example: Converting a Mixed-Case String to Uppercase

UPCASE converts the LAST_NAME_MIXED field to uppercase:

```
DEFINE FILE EMPLOYEE
LAST_NAME_MIXED/A15=IF DEPARTMENT EQ 'MIS' THEN LAST_NAME ELSE
  LCWORD(15, LAST_NAME, 'A15');
LAST_NAME_UPPER/A15=UPCASE(15, LAST_NAME_MIXED, 'A15') ;
END
```

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME_MIXED AND FIRST_NAME BY LAST_NAME_UPPER
WHERE CURR_JOBCODE EQ 'B02' OR 'A17' OR 'B04';
END
```

Now, when you execute the request, the names are sorted correctly.

The output is:

LAST_NAME_UPPER	LAST_NAME_MIXED	FIRST_NAME
-----	-----	-----
BANNING	Banning	JOHN
BLACKWOOD	BLACKWOOD	ROSEMARIE
CROSS	CROSS	BARBARA
MCCOY	MCCOY	JOHN
MCKNIGHT	Mcknight	ROGER
ROMANS	Romans	ANTHONY

If you do not want to see the field with all uppercase values, you can NOPRINT it.

Creating Multi-Locale Applications

WebFOCUS allows you to display valuable information in a meaningful way. It is not difficult to provide your users with information in the language of their choice. This appendix describes some ways to structure a multi-locale applications without the need to create a separate file for each language.

In this appendix:

- ☐ [Creating a Multi-Locale Application Using LNGPREP](#)
 - ☐ [Displaying Multilingual Reports](#)
 - ☐ [Localizing Portals](#)
-

Creating a Multi-Locale Application Using LNGPREP

In this example, you will use the LNGPREP file to create language files that you can translate, allowing you to set a parameter to change the language of a procedure on-demand at run-time. This procedure uses the WebFOCUS Reporting Server Console to create the .cfg and .lng configuration files, and WebFOCUS App Studio to create the parameterized report. You will follow the following steps:

- ☐ Create a .cfg file to specify the language in which to localize the Master File.
- ☐ Use the LNGPREP function to generate a .lng file for each of these languages and add translations to each one.
- ☐ Create a new procedure that provides single-select lists in the Responsive Autoprompt facility, allowing you to switch between localized reports on-demand.
- ☐ Test the project.

Procedure: **How to Create Multi-Locale Master Files**

You can use the LNGPREP function in a procedure to create localization files for any synonym on your Reporting Server. This function references a .cfg language input file that

lists the languages that you wish to localize for. As a result, this language input file must be created first.

1. In the Web Console, locate the *ggsales* synonym. If you do not have it already, you can generate it.
 - a. The *ggsales* synonym is included in the Legacy Sample Tables and Files. To create these files on your environment, in the Applications tab of the Web Console, right click the application folder that you wish to use (for example, you could use a new application folder called *nls*), point to *New*, and click *Tutorials*.

The Create Tutorial Framework page opens.

- b. From the Tutorial drop-down menu, select *Create Legacy Sample Tables and Files* and click *Create*.

The Select an Option dialog box opens. Click *OK*.

A number of sample synonyms are created, including *ggsales*.

2. Create the language input file. This is a *.cfg* file that contains a list of all of the languages that you want to localize your synonym for. For this example, the language input file will specify that localizations are being created for French, Spanish, and Portuguese.

- a. Right-click the application folder, point to *New*, and click *Text File*.

The text editor opens in the Web Console.

- b. The *.cfg* file is formatted with one language per line, as three-character codes, with no punctuation. To specify French, Spanish, and Portuguese as the localized languages, type the following in the text file:

```
fre  
spa  
por
```

A table containing language codes can be found in [How to Specify Multilingual DESCRIPTION and TITLE Attributes](#) on page 146.

Note: English (*eng*) does not need to be specified. It is used as the base value for LNGPREP, even if the original file is not in English. If you are localizing a Master File from a foreign language to English, you must use a more specific English code, like *uke* for British English or *ame* for American English, to specify an English localization.

- c. Once you have entered the languages you are localizing for, click *Save As* in the text editor. Give the file a name and type *cfg* in the Extension text box to save the file as a *.cfg* file.
3. Create the *.lng* translation files. These files will be translated to provide localized column and table names.
 - a. In the navigation pane, right-click the *ggsales* synonym, point to *Metadata Management*, and click *Prepare Translation Files*.

The Set Translation Files page opens.

- b. Enter or browse for the application folder in which to save the translation files. This folder can be within a hierarchy, such as `nls/ggsales`.
- c. Set a prefix that will be included in the name of each translation file. For example, if you set the prefix to `ggsales_`, and have *fre*, *spa*, and *por* in the language input file, then the translation files `ggsales_eng.lng`, `ggsales_fre.lng`, `ggsales_spa.lng`, and `ggsales_por.lng` will be generated.

As noted previously, `prefixeng.lng` is generated automatically and serves as the base translation file containing the column and table names specified in the Master File, even if those names are not in English. You would use the `prefixame.lng` or `prefixuke.lng` file as the English translation file.

- d. Specify the `.cfg` language input file by entering or browsing for it in the Languages File field.
- e. Click *OK*.

You are informed that the translation files have been created in the specified folder.

4. Refresh the navigation pane and open each translation file, except for the base `prefix_eng` translation file, to translate it. Translate each title and description in the translation file. Do not modify the numbers in the file.

Each translation file matches a number with each title and description in the Master File. The text next to each number must represent the same fields and segments in each translation file for the translations to be correct.

You can translate the values in the translation files for `ggsales`, as follows:

<code>ggsales_eng.mas</code>	<code>ggsales_fra.mas</code>	<code>ggsales_spa.mas</code>	<code>ggsales_por.mas</code>
(Base Values)			
Legacy Metadata Sample: Gotham Grinds - GGSales	Exemple Métadonnées héritées: Gotham Grinds - GGSales	Ejemplo metadatos heredados: Gotham Grinds - GGSales	Exemplo de Metadados de Legado: Gotham Grinds - GGSales
Sequence#	# de séquence	n.º de secuencia	Sequência#
Sequence number in database	Nombre de séquence dans Base de données	Número de secuencia en base de datos	Número de sequência no banco de dados

ggsales_eng.mas	ggsales_fra.mas	ggsales_spa.mas	ggsales_por.mas
(Base Values)			
Category	Catégorie	Categoría	Categoria
Product category	Produit Catégorie	Categoría de Producto	Produto Categoria
Product ID	ID de produit	ID de Producto	ID do produto
Product Identification code (for sale)	Code d'identification du produit (en vente)	Código de identificación del producto (para la venta)	Código de identificação do produto (para vendas)
Product	Produit	Producto	Produto
Product name	Nom de Produit	Nombre de Producto	Nome do produto
Region	Région	Región	Região
Region Code	Code de région	Código de región	Código da região
State	État	Estado	Estado
City	Ville	Ciudad	Cidade
Store ID	ID de magasin	ID de tienda	ID da loja
Store identification code (for sale)	Code d'identification du magasin (en vente)	Código de identificación de la tienda (para la venta)	Código de identificação da loja (para vendas)
Date	Date	Fecha	Data
Date of sales report	Date du rapport de vente	Fecha del informe de venta	Relatório de data de vendas
Unit Sales	Ventes en unités	Ventas de unidades	Vendas por unidade

ggsales_eng.mas	ggsales_fra.mas	ggsales_spa.mas	ggsales_por.mas
(Base Values)			
Number of units sold	Nombre d'unités vendues	Número de unidades vendidas	Número de unidades vendidas
Dollar Sales	Ventes en dollars	Ventas en dólares	Vendas em dólares
Total dollar amount of reported sales	Montant total des ventes en dollars	Volumen total de ventas en dólares	Quantidade total em dólares de vendas relatadas
Budget Units	Unités budgétaires	Unidades presupuestadas	Unidades de orçamento
Number of units budgeted	Nombre d'unités budgétées	Número de unidades presupuestado	Número de unidades orçadas
Budget Dollars	Budget en dollars	Presupuesto en dólares	Orçamento em dólares
Total sales quota in dollars	Quota total des ventes en dollars	Cuota total de ventas en dólares	Cota total de vendas em dólares

Once you have translated each translation file as described, save each one.

Once complete, the Master File should have the TRANS_FILE parameter automatically added to it. To check, right-click the Master File and click *Edit as Text*. The parameter should appear on the first line, listing the location and prefix of the translation files.

Procedure: How to Create a Procedure for Multi-Locale Reporting

Once your data has been localized, it is easy to use in a procedure, such as a report.

1. In App Studio, create a new report based on the ggsales synonym.
 - a. In the Environments Tree panel, right-click the folder where you want to create the report, point to *New*, and click *Report*.
 - b. In the Select Data Source dialog box, click the application folder containing the ggsales synonym and open *ggsales.mas*.

The Report canvas opens.

2. Add the desired fields to your report, and save and close it.
3. Next, create parameters to set the language, date format, and decimal notation of the report based on user selection in the Responsive Autoprompt facility.

These parameters are created as the values for SET commands. You can create a SET command by typing it above the TABLE FILE request in your report.

For more information on SET commands, see the *Developing Reporting Applications* manual.

Note that all set commands should be kept to one line. Any line breaks in this section are due to size constraints on this page.

- a. In the Environments Tree panel, right-click your newly created report and click *Open in Text Editor*.
- b. At the top of the report, create a simple parameter for LANG that uses a single-select list.

LANG sets the language of the report. For more information, see [How to Activate the Use of a Language](#) on page 150.

The syntax for creating a single-select static list for LANG is as follows:

```
SET LANG = '&variable.  
( <displaylang1,codelang1>,<displaylang2,codelang2>, ... ).Title.'
```

where:

&variable

Is the name of the variable, including the ampersand (&), for which you are supplying a list of values.

displaylang1

Is the display value for the first language that will display in the Responsive Autoprompt drop-down menu.

codelang1

Is the three-character language code for the first language that will display in the Responsive Autoprompt drop-down menu. This value is also found in the title of the translation file.

displaylang2

Is the display value for the first language that will display in the Responsive Autoprompt drop-down menu.

code lang2

Is the three-character language code for the first language that will display in the Responsive Autoprompt drop-down menu. This value is also found in the title of the translation file.

...

The display name and language code values for any other options in the drop-down menu, separated by a comma and wrapped in angle brackets.

Title

Is the title that will display above the drop-down menu.

For this example, you can use the following command:

```
SET LANG = '&Lang.  
( <English,eng>,<French,fre>,<Spanish,spa>,<Portuguese,por> )  
.Language.'
```

- c. Create another simple parameter for the date format, if you are using a date field in your report. The date format is set using the DATE_ORDER command.

The syntax to create a simple parameter with a single-select list for DATE_ORDER is as follows:

```
SET DATE_ORDER = '&variable.(DMY,MDY,YMD).Title.'
```

where:

&variable

Is the name of the variable, including the ampersand (&), for which you are supplying a list of values.

DMY,MDY,YMD

Are the available options to use as values for DATE_ORDER. DMY represents day/month/year, MDY represents month/day/year, and YMD represents year/month/day.

Title

Is the title that will display above the drop-down menu.

Note that display values are not being specified. In this case, the possible DATE_ORDER values are displayed literally in the Responsive Autoprompting drop-down menu.

For this example, you can use the following command:

```
SET DATE_ORDER = '&Date_order.(DMY,MDY,YMD).Date Order.'
```

- d. Create a simple parameter for decimal notation, which determines the formatting for decimals and large numbers. Decimal notation is set by the CDN command. For more information, see [Punctuating Large Numbers](#) on page 129.

The syntax to create a simple parameter with a single-select list for CDN is as follows:

```
SET CDN = '&variable.  
(<displayoff,OFF>,<displayon,ON>,<displayspace,SPACE>,  
<displayquote,QUOTE>,<displayquotep,QUOTEP>).Title.'
```

where:

&variable

Is the name of the variable, including the ampersand (&), for which you are supplying a list of values.

displayoff

Is the value to display in the menu for when CDN is set to OFF.

OFF

Is the default format. Alternatively, COMMAS_DOT serves the same function. Commas are used as thousands separators and a period is used as the decimal point. For example, 123,456.78.

displayon

Is the value to display in the menu for when CDN is set to ON.

ON

Turns CDN on. Alternatively, DOTS_COMMA serves the same function. Periods are used as thousands separators and a comma is used as the decimal point. For example, 123.456,78.

displayspace

Is the value to display in the menu for when CDN is set to SPACE.

SPACE

Sets CDN to SPACE. Alternatively, SPACES_COMMA serves the same function. Spaces are used as thousands separators and a comma is used as the decimal point. For example, 123 456,78.

displayquote

Is the value to display in the menu for when CDN is set to QUOTE.

QUOTE

Sets CDN to QUOTE. Alternatively, QUOTES_COMMA serves the same function. Quotes are used as thousands separators and a comma is used as the decimal point. For example, 123'456,78.

displayquotep

Is the value to display in the menu for when CDN is set to QUOTEP.

QUOTE

Sets CDN to QUOTE. Alternatively, QUOTES_DOT serves the same function. Quotes are used as thousands separators and a period is used as the decimal point. For example, 123'456.78.

Title

Is the title to show above the drop-down menu in Responsive Autoprompt.

For this example, you can use the following command:

```
SET CDN = '&CDN.(<Off,OFF>,<On,ON>,<Spaces with Comma,SPACE>,<Quotes with Comma,QUOTE>,<Quotes with Dot,QUOTE>).Decimal Notation.'
```

The following is an example of a full report.

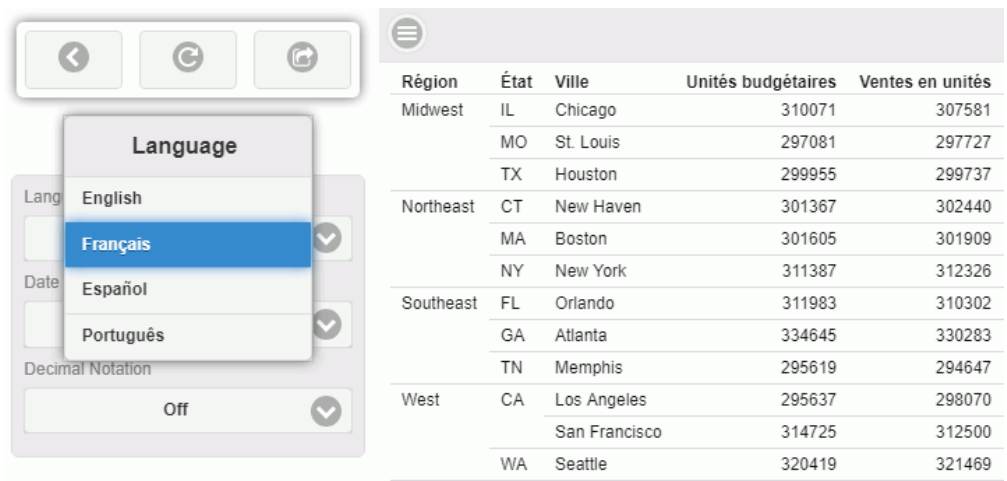
```
SET LANG = '&Lang.
(<English,eng>,<Français,fre>,<Español,spa>,<Português,por>).Language.'
SET DATE_ORDER = '&Date_order.(DMY,MDY,YMD).Date Order.'
SET CDN = '&CDN.(<Off,OFF>,<On,ON>,<Spaces with Comma,SPACE>,<Quotes with Comma,QUOTE>,<Quotes with Dot,QUOTE>).Decimal Notation.'
```

```
TABLE FILE nls/GGSALES
SUM
    GGSALES.SALES01.BUDUNITS
    GGSALES.SALES01.UNITS
    GGSALES.SALES01.DOLLARS
BY  GGSALES.SALES01.REGION
BY  GGSALES.SALES01.ST
BY  GGSALES.SALES01.CITY
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTML EMBEDDING ON
ON TABLE SET HTML CSS ON
ON TABLE SET STYLE *
    INCLUDE = IBFS:/EDA/EDASERVE/_EDAHOME/ETC/warm.sty,
$
ENDSTYLE
END
```

4. Save and run the report.

You are prompted to select a language, date order, and decimal notation from the Responsive Autoprompt interface.

Select your options and click the run button. The report loads to match your specifications, as shown in the following image.



Displaying Multilingual Reports

To display data in different languages, you must do one of the following:

- ❑ Create a different data source for each language.
- ❑ Create one data source for all languages, if they share a common code page. With this technique, each record should contain a key field indicating the language. A report request tests the key field to retrieve a record in the chosen language.

In either case, if you are using a language that has different alphabets (such as Japanese), make sure that the WebFOCUS Reporting Server and WebFOCUS Client are configured for code page 65001 (Unicode).

You do not need to store most kinds of data in different languages. For example, names, addresses, and numerical data do not require translation. In the case of numerical data, you can easily manage the display format used by the locale at the procedure level.

This appendix describes the steps for creating an application that displays page headings and column titles of a report in different languages, based on a user selection. The following illustrates the reports, in four languages, that the user can request. Notice that the dates and numerical data are formatted according to the conventions of the locale.

Proceed to the next topic for instructions on creating this application.

You can also create a localized application using the LNGPREP function as shown in the following image. For more information, see [Creating a Multi-Locale Application Using LNGPREP](#) on page 221.

Language
English
▶ ◀

Movie List

Movie Code	Title	Category	Director	Rating	RELDATE	Wholesale Price	List Price	Copies
001MCA	JAWS	ACTION	SPIELBERG S.	PG	05/13/78	10.99	19.95	2

005
Language
Español
▶ ◀

Lista de Cine

Código Cine	Título	Categoría	Director	Indice	RELDATE	Costo	Precio	Copia
001MCA	JAWS	ACTION	SPIELBERG S.	PG	13/05/78	10,99	19,95	2

005
Language
Français
▶ ◀

Liste des films

Code du Film	Titre	Catégorie	Classification	Classification	RELDATE	Coût	Prix	Copies
001MCA	JAWS	ACTION	SPIELBERG S.	PG	13/05/78	10,99	19,95	2

005
Language
Português
▶ ◀

Lista de filme

Código do filme	Título	Categoria	Diretor	Censura	RELDATE	Preço de atacado	Preço de lista	Cópias
001MCA	JAWS	ACTION	SPIELBERG S.	PG	13-05-78	10,99	19,95	2
005WAR	EAST OF EDEN	CLASSIC	KAZAN E.	NR	12-01-55	14,99	24,98	1

How Does the Sample Application Work?

A single, integrated procedure produces the application launch page and the report in different languages. You will mainly use two graphical development tools in App Studio—the Report canvas and the HTML canvas. The Report canvas creates and styles the report, which accesses the MOVIES data source, included in the Legacy Sample Tables and Files tutorial, which you can generate in the WebFOCUS Reporting Server Console. The HTML canvas creates the launch page and links it to the report.

In the sample application, you will prompt the user to select English, French, Spanish, or Portuguese from a drop-down list. The choice corresponds to an amper variable (&LNG) in the procedure. The variable &LNG determines the language of the report.

The procedure will look similar to the following when you are done.

```
-DEFAULT &LNG = 'EN';
-SET &HEAD = IF &LNG EQ 'EN' THEN 'Movie List' ELSE
- IF &LNG EQ 'FR' THEN 'Liste des films' ELSE
- IF &LNG EQ 'ES' THEN 'Lista de Cine' ELSE
- IF &LNG EQ 'PT' THEN 'Lista de filme';
-SET &FMT = IF &LNG EQ 'EN' THEN '/MDY' ELSE
- IF &LNG EQ 'FR' THEN '/DMY' ELSE
- IF &LNG EQ 'ES' THEN '/DMY' ELSE
- IF &LNG EQ 'PT' THEN '/D-M-Y';
-IF &LNG EQ 'EN' GOTO BEGIN;
SET CDN = DOTS_COMMA
-BEGIN;
USE
nls/movies.foc AS MOVIES_&LNG
END
TABLE FILE MOVIES_&LNG
SUM
    WHOLESALEPR AND
    LISTPR AND
    COPIES
BY MOVIECODE
BY TITLE
BY CATEGORY
BY DIRECTOR
BY RATING
BY RELDATE&FMT
HEADING
"&HEAD"
ON TABLE SET PAGE-NUM NOLEAD
ON TABLE SET ASNAMES ON
ON TABLE NOTOTAL
ON TABLE PCHOLD FORMAT HTML
ON TABLE SET HTMLEMBEDIMG ON
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
    INCLUDE = warm,
$
ENDSTYLE
END
```

The column titles of the fields are translated in .lng files, which are generated for each language listed in a .cfg file. This enables you to create procedures localized in a variety of languages from a single Master File. In this example, the movies.mas Master File is localized for French, Spanish, and Portuguese.

The language, decimal notation, and date format are determined by SET commands and controlled by amper variables in the procedure.

WebFOCUS provides many features that make it easy to configure National Language Support (NLS) at your enterprise. Because you translate the .lng files yourself and can generate them for any language for which there is a three-character language code, you are not limited to languages for which there is a WebFOCUS localized version. It is easy to produce many different versions of the same report to accommodate international users.

Development Steps

In the example, you will perform the following tasks:

- ☐ Create four synonyms for different languages pointing to the same data source.
- ☐ Create a new procedure that provides a report in one of four languages, depending on the user selection.
- ☐ Create a launch page that prompts the user for the language of choice.
- ☐ Test the project.

Procedure: How to Create Multi-Locale Master Files

You will create a Master File for each of the four languages that will be available in the application (English, French, Spanish, Portuguese) based on the Movies data source.

1. In the Web Console, locate the movies synonym. If you do not have it already, you can generate it. If you do have it, you can proceed to step 2.
 - a. The movies synonym is included in the Legacy Sample Tables and Files. To create these files on your environment, in the Applications tab of the Web Console, right-click the application folder that you wish to use (for example, you could use a new application folder called *nls*), point to *New*, and click *Tutorials*.

The Create Tutorial Framework page opens.

- b. From the Tutorial drop-down menu, select *Create Legacy Sample Tables and Files* and click *Create*.

The Select an Option dialog box opens. Click *OK*.

A number of sample synonyms are created, including movies.

2. Create four copies of the movies.mas synonym. Each will be used to provide the column titles for one language.
 - a. In the Applications tab of the Web Console, right-click the movies.mas synonym. The text editor opens, showing the syntax of the movies.mas Master File.
 - b. Specify the DATASET attribute for the Master File. This will specify the data that will be used with this Master File, ensuring that each localized copy uses the same source data.

On first line of syntax in the Master File, just before the dollar sign that is used to end the line, type the following:

```
DATASET = app/movies.foc,
```

where:

app

Is the application folder containing the movies.foc data source. This file was created when you generated the Legacy Sample Tables and Files tutorial.

The full first line of the Master File should now resemble the following:

```
FILENAME=MOVIES, SUFFIX=FOC, REMARKS='Legacy Metadata Sample: movies  
(for use with Video Track example)', DATASET= app/movies.foc,$
```

- c. In the text editor, click **Save As** and save the file as *moviesEN.mas* to signify that this is the English language version of the Master File.
- d. Repeat step c three more times, but save the Master File as *moviesES.mas*, *moviesFR.mas*, and *moviesPT.mas*.

Your application folder should now contain the following Master Files, all of which are identical:

- ☐ moviesEN.mas
- ☐ moviesES.mas
- ☐ moviesFR.mas
- ☐ moviesPT.mas

- e. Exit the text editor by closing the tab, but stay in the Applications tab of the Web Console.
3. For each field in each Master File, supply a column title based on the table shown below.

Each column title should be in the language of the associated Master File.

- a. Right-click the *moviesEN.mas* synonym and click *Open*. Data Assist opens.
- b. Click *Create Default* to generate the default Business View of your fields.
- c. Rename each field as shown in the table below, then save the synonym.

To rename a field, right-click the name in the Display Name (Title) column of the Business View pane, click *Rename*, and type in a new name.

Once you have renamed all fields, open the File menu on the ribbon and click **Save** to save your changes.

Repeat these steps for each of the four Master Files.

Field Name	moviesEN.mas	moviesES.mas	moviesFR.mas	moviesPT.mas
MOVIECODE	Movie,Code	Codigo,Cine	Code du,Film	Código,do filme
TITLE	Title	Título	Titre	Título
CATEGORY	Category	Categoria	Catégorie	Categoria
DIRECTOR	Director	Director	Directeur	Diretor
RATING	Rating	Indice	Classification	Censura
RELDATE	Release,Date	Fecha de,Estreno	Date de,Parution	Data de,produção
WHOLESALEPR	Wholesale,Price	Costo	Coût	Preço de,atacado
LISTPR	List,Price	Precio	Prix	Preço,de lista
COPIES	Copies	Copia	Copies	Cópias

Once complete, each Master File should now have a TITLE attribute showing the translated column title for each field. You can check by opening each of the modified Master Files as text.

Procedure: How to Create a Procedure for Multi-Locale Reporting

Once your data has been localized, it is easy to use in a procedure, such as a report.

1. In App Studio, create a new report based on the movies.mas synonym.
 - a. In the Environments Tree panel, right-click the folder where you want to create the report, point to *New*, and click *Report*.
 - b. In the Select Data Source dialog box, click the application folder containing the movies synonym and open *movies.mas*.
The Report canvas opens.
2. Add a Use command to associate the data source (movies.foc) to the procedure.
 - a. In the Procedure View panel, right-click the top component (the Comment component by default), point to *New*, and click *Use*. The Use component should appear above the Report component.
 - b. Click the *Browse...* button for Database Filename, and navigate to and select the movies.foc data source in your application directory.

- c. In the MASTER File text box, type *MOVIESEN*.
 - d. Click *Add* to add the data source.
3. Add a Set command to display numeric data with a comma as the decimal separator in the French, Portuguese, and Spanish reports. Another command will be added later so that the English report ignores this section.
 - a. In the Procedure View panel, right-click the top component (the Comment component by default), point to *New*, and click *Set*. The Set component should appear above the Use component.
 - b. From the Available Settings list box, select *CDN* and click *Add*. Set the Current Value to *DOTS_COMMA*, which is the format of continental decimal notation.
 - c. Return to the Report canvas.
4. Add every field listed under MOVINFO in the Object Inspector to the report, and style the report however you prefer.
5. Add the page heading *Movie List* to the report.

On the Report tab of the ribbon, in the Report group, click *Header & Footer* and click *Page Header*. Type *Movie List* into Page Heading text box.
6. Add a command to allow you to skip the Set command for the English version of the application.
 - a. Right-click the Set command, point to *New*, and click *Other*.
A text editor tab opens.
 - b. Type in the following text:

```
-BEGIN;
```
7. Save and close the report.
8. Reopen the report in a text editor to make additional modifications.

In the Environments Tree panel, right-click the report and click *Open in Text Editor*. The report syntax is visible in a text editor in App Studio.
9. The fields must be referenced by the fieldname only. If the fields are referenced by a qualified field name, for example, MOVIES.MOVINFO.MOVIECODE, you must change it to just *MOVIECODE* so that they match the field names in the localized Master Files. Remove *MOVIES.MOVINFO* from any field names in the report request.
10. Add the following syntax to the top of the procedure to include the amper variable &LNG for the language of the report, &HEAD for the page heading, and &FMT for the format of the report. This text should go above the SET CDN command.

```

-DEFAULT &LNG = 'EN';
-SET &HEAD = IF &LNG EQ 'EN' THEN 'Movie List' ELSE
- IF &LNG EQ 'ES' THEN 'Lista de Cine' ELSE
- IF &LNG EQ 'FR' THEN 'Liste des films' ELSE
- IF &LNG EQ 'PT' THEN 'Lista de filme';
-SET &FMT = IF &LNG EQ 'EN' THEN '/MDY' ELSE
- IF &LNG EQ 'ES' THEN '/DMY' ELSE
- IF &LNG EQ 'FR' THEN '/DMY' ELSE
- IF &LNG EQ 'PT' THEN '/D-M-Y';
-IF &LNG EQ 'EN' GOTO BEGIN;

```

This syntax performs the following functions:

- ☐ Sets the default language to English. If you do not make a language selection when running the final application, English will be used.
 - ☐ Creates an amper variable to determine the page heading based on the selected language.
 - ☐ Creates an amper variable to determine the date format based on the selected language.
 - ☐ Tells the procedure to skip the SET CDN command if English is the selected language.
11. Change *MOVIESEN* to *MOVIES&LNG* in the syntax so that making a language selection determines the Master File that is referenced.

The line under USE should now look like this:

```
app/movies.foc AS MOVIES&LNG
```

where

app

Is the application folder containing movies.foc. This should already be defined in the report syntax.

and the TABLE FILE declaration should be:

```
TABLE FILE MOVIES&LNG
```

12. Change the heading from *Movie List* to *&HEAD* so that the localized heading defined in the first -SET command is used.

The HEADING block of the report syntax should look like the following:

```
HEADING
"&HEAD"
```

13. Add *&FMT* to the end of the RELDATE field name to change the date format based on the selected language.

The line containing RELDATE should now look like the following:

`BY RELDATE&FMT`

14. Save and close the report.

You are now ready to create an HTML page with a control to set the &LNG parameter in the report.

Procedure: How to Create a Launch Page for Multi-Locale Reporting

You will use an HTML page as a launch page for the report where you can select the language that the report will display.

1. Create a new HTML page.
 - a. In the Environments Tree panel, right-click a folder, point to *New*, and click *HTML/Document*, or click *HTML/Document* in the Content group on the Home tab on the ribbon.
 - b. Proceed through the HTML/Document Wizard, selecting the settings of your choice.
2. In the HTML canvas, create a report container. This is where the report will display when you run the page.

On the ribbon, on the Components tab, in the Reports group, click *Report*. Drag your mouse across the page to draw the report container.

3. Add the report to container.

Right-click the report container and click *Reference existing procedure*. Navigate to and select the report created in [How to Create a Procedure for Multi-Locale Reporting](#) on page 235.

You are alerted that there are undefined parameters in the report. Close any warnings until the New Parameters dialog box displays.

4. In the New Parameters dialog box, create the controls for the page.

Right-click the value in the Control Type column to change it to *Drop Down List* if it is not one already. This will allow you to specify the available menu options.

Click *OK* to create the control.

You can optionally create your own controls using the Controls tab on the ribbon and using the Settings panel to link the control to the &LNG parameter. For more information, see the *WebFOCUS App Studio User's Manual*.

A group box containing the controls appears on the page. You can modify any of the elements that it contains, such as resizing any of the components.

5. Change the display values for the drop-down menu to make it more user friendly.

By default, this menu only shows EN, since this was set as the default. You can add the other values to the menu and set the display values to show the names of each language.

- a. Select the drop-down menu object on the page.
- b. Open the *Settings* panel. EN is the only value, since it was defined as the default.
- c. Set the Display to *English* so that the name of the language will appear in the menu but set the &LNG value to *EN*.
- d. Above the list of values, click the *New* icon.
- e. Add the menu option for the Spanish version of the report. Set the Value to *ES* to match the value for &LNG and set the Display to *Español*.
- f. Add the menu option for the French version of the report. Set the Value to *FR* to match the value for &LNG and set the Display to *Français*.
- g. Add the menu option for the Portuguese version of the report. Set the Value to *PT* to match the value for &LNG and set the Display to *Português*.

The drop-down menu is now configured.

6. Make any further modifications that you wish to the page, then once you are done, save it.

Your page is complete. When you run the page, making a selection from the drop-down menu and clicking the run button on the HTML page should reload the report in the selected language, with localized column titles, date format, and page heading.

Localizing Portals

WebFOCUS is a fully internationalized platform that supports the Unicode/UTF-8 standard, enabling operating systems, browsers, and applications to process and display information in virtually any spoken or written language. In addition to these default localization capabilities, users can also localize their own interface customizations. You can localize custom components in WebFOCUS in one of two ways:

- ☐ By implementing symbolic references.
- ☐ By configuring a translation file.

Note: This option is only available for basic portals. It is deprecated in favor of the symbolic references option.

Procedure: How to Localize Custom Components Using a Translation File

1. On the Legacy Home Page, in the Resources tree, right-click a basic portal, and then click *Translations*.

The Translations file opens. The syntax contains the identification number for every component in the portal, its name, and language reference, as shown in the following example.

```
<translations>
  <viewid value="EMBEDDED_PORTAL_1-1WEIVNL07X8" />
  <translation id="EMBEDDED_PORTAL_1-1WEIVNL07X8" language="en_US"
key="name" value="Embedded Portal 1"/>
  <translation id="VIEW-3U1SPSUGZUREZ" language="en_US"
key="description" value=""/>
  <translation id="PAGE1-JB1X909F5FGX9" language="en_US" key="name"
value="Page 1"/>
  <translation id="PAGE1-JB1X909F5FGX9" language="en_US"
key="description" value=""/>
  <translation id="CONTAINER1-JT1SO4V1I13KL" language="en_US"
key="name" value="Panel 1"/>
  <translation id="CONTAINER2-JZ156JV9OZQM7" language="en_US"
key="name" value="Product Line Sales"/>
  <translation id="PANEL11-K41N5OL9ZDRPS" language="en_US" key="text"
value="By Product"/>
  <translation id="PANEL8-KD1A7FGQK9F7R" language="en_US" key="text"
value="Overall"/>
  <translation id="PANEL9-KS1UQOXUF2L3A" language="en_US" key="text"
value="By Regional"/>
  <translation id="TITLEBAR2-L612GN2S7S1CZ" language="en_US"
key="text" value="Product Line Sales"/>
  <translation id="TITLEBAR2-L612GN2S7S1CZ" language="en_US"
key="text_base" value="Panel 2"/>
</translations>
```

2. Copy and paste the sections of syntax that you want to translate, and account for as many translations as you need. For example, if you want to translate a basic portal to Spanish, you would only copy and paste the syntax once. If you want to translate a basic portal to Spanish and French, you would copy and paste the syntax twice.

In this example, we only use English and Spanish.

3. In the copied portion of the syntax, change every instance of *en_US* to the language that you want to translate. In this example, we use *es_ES* for Spanish.
4. In the copied portion of the syntax, change the name of every component that appears in quotes after the word *value*, to the equivalent word in the language you are translating.

The modified syntax should look similar to the following:


```

<translations>
<viewid value="EMBEDDED_PORTAL_1-1WEIVNL07X8" />
  <translation id="EMBEDDED_PORTAL_1-1WEIVNL07X8" language="en_US"
key="name" value="Embedded Portal 1"/>
  <translation id="VIEW-3U1SPSUGZUREZ" language="en_US"
key="description" value=""/>
  <translation id="PAGE1-JB1X909F5FGX9" language="en_US" key="name"
value="Page 1"/>
  <translation id="PAGE1-JB1X909F5FGX9" language="en_US"
key="description" value=""/>
  <translation id="CONTAINER1-JT1SO4V1I13KL" language="en_US"
key="name" value="Panel 1"/>
  <translation id="CONTAINER2-JZ156JV9OZQM7" language="en_US"
key="name" value="Product Line Sales"/>
  <translation id="PANEL11-K41N50L9ZDRPS" language="en_US" key="text"
value="By Product"/>
  <translation id="PANEL8-KD1A7FGQK9F7R" language="en_US" key="text"
value="Overall"/>
  <translation id="PANEL9-KS1UQOXUF2L3A" language="en_US" key="text"
value="By Regional"/>
  <translation id="TITLEBAR2-L612GN2S7S1CZ" language="en_US"
key="text" value="Product Line Sales"/>
  <translation id="TITLEBAR2-L612GN2S7S1CZ" language="en_US"
key="text_base" value="Panel 2"/>
<translation id="EMBEDDED_PORTAL_1-1WEIVNL07X8" language="es_ES"
key="name" value="Portal incrustado 1"/>
  <translation id="VIEW-3U1SPSUGZUREZ" language="es_ES"
key="description" value=""/>
  <translation id="PAGE1-JB1X909F5FGX9" language="es_ES" key="name"
value=" Página 1"/>
  <translation id="PAGE1-JB1X909F5FGX9" language="es_ES"
key="description" value=""/>
  <translation id="CONTAINER1-JT1SO4V1I13KL" language="es_ES"
key="name" value="Panel 1"/>
  <translation id="CONTAINER2-JZ156JV9OZQM7" language="es_ES"
key="name" value=" Línea de productos Ventas "/>
  <translation id="PANEL11-K41N50L9ZDRPS" language="es_ES" key="text"
value=" Por Producto "/>
  <translation id="PANEL8-KD1A7FGQK9F7R" language="es_ES" key="text"
value=" En general "/>
  <translation id="PANEL9-KS1UQOXUF2L3A" language="es_ES" key="text"
value=" Por Regional "/>
  <translation id="TITLEBAR2-L612GN2S7S1CZ" language="es_ES"
key="text" value=" Línea de productos Ventas "/>
  <translation id="TITLEBAR2-L612GN2S7S1CZ" language="es_ES"
key="text_base" value="Panel 2"/>
</translations>

```

5. Optionally, repeat steps 2 through 4 to add a translation to another language.
6. Save and close the Translations file.

Your translation is complete.

Procedure: How to Localize Custom Components Using Symbolic References

This procedure requires a Java development tool, such as Eclipse, NetBeans, or Java Development Kit (JDK). In this example, we are using Eclipse Java EE IDE.

1. If using Eclipse, follow the steps below:

- a. Launch Eclipse.
- b. Click *File*, point to *New*, and then click *Java Project*.

The New Java Project dialog box opens.

- c. In the Project name field, type the name of your project and click *Finish*.
- d. Right-click your new project, point to *New*, and then click *Package*.

The New Java Package dialog box opens.

- e. In the Name field, type the following name:

```
com.ibi.desired_name.intl.templates
```

where:

```
desired_name
```

Is any identifiable name that you want to use for your localized data.

- f. Click *Finish*.
- g. Right-click the package that you just created, point to *New* and then click *File*.

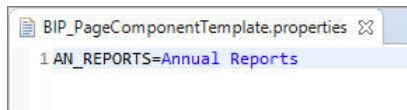
The New File dialog box opens.

- h. In the File Name field, type *BIP_PageComponentTemplate.properties* and click *Finish*.
- i. Double-click the file that you just created.

The editor opens in the right pane.

- j. Type a symbolic reference of your choice, followed by the equal sign, and the value for that symbolic reference.

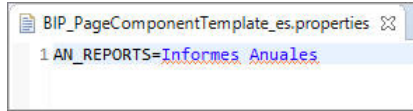
An example of the simple symbolic reference syntax is shown in the following image.



- k. Repeat Steps g and h to create a PROPERTIES file for a different language. In this example, we create a file for the Spanish translation and name it *BIP_PageComponentTemplate_es.properties*.
- l. Double-click the PROPERTIES file for the language you are translating.

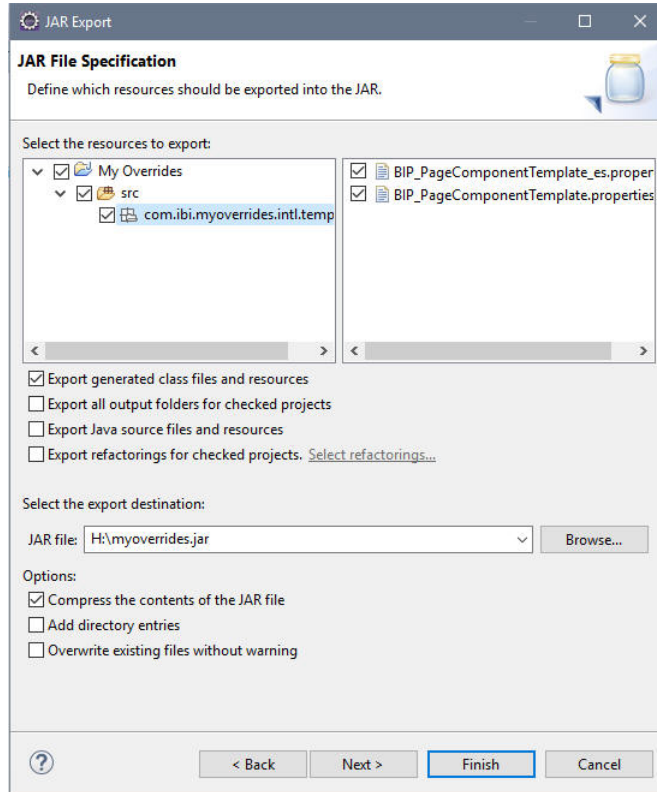
- m. Type the same symbolic reference as you did in the `BIP_PageComponentTemplate.properties` file, followed by the equal sign (=), and the Spanish translation of the value.

An example of the Spanish version of the syntax is shown in the following image.



- n. Save and close both `PROPERTIES` files.
- o. Right-click the project and then click *Export*.
The Export wizard opens.
- p. Under the Java folder, select *JAR file* and click *Next*.
The JAR Export window opens.
- q. Make sure all your elements, such as the project, package, and translated `PROPERTIES` files, are selected, and click *Browse*.
The Save As dialog box opens.
- r. Point to a location in your environment where you want to export the JAR file.
- s. In the File name field, type the same custom name that you used as part of the package name and click *Save*.

An example of a completed JAR export window is shown in the following image.



- t. Click *Finish* and close Eclipse.
2. If using JDK, follow the steps below:
 - a. In your environment, create the following directory structure:


```
com\ibi\desired_name\intl\templates
```

where:

desired_name

Is any identifiable name that you want to use for your localized data.
 - b. Make sure there is one higher level directory above the com folder. In our example, the higher level directory is called Localization.
 - c. Inside the templates folder, create the BIP_PageComponentTemplate.properties file and its versions for the languages that you want to use.
 - d. Create custom symbolic references inside each PROPERTIES file, as described in step 1.

- e. Save and close PROPERTIES files.
- f. Open the Command Prompt.
- g. Type the following string:

```
Disk:\PROGRA~1\path to JDK\jar cvf jar-file-name.jar -C higher
level directory name/ .
```

where:

Disk:

The disk where the JDK is located.

path to JDK

The path to the JDK inside your Program Files directory.

jar-file-name.jar

The name of your JAR file. Use the same name that you used as part of the directory structure in Step a.

higher level directory name

The name of the directory where the com folder resides.

- h. Press Enter.

The JAR file is created.

The following image shows an example of a completed command.

```

C:\> Command Prompt
H:\c:\PROGRA~1\java\jdk1.8.0_121\bin\jar cvf myoverrides.jar -C Localization/ .
added manifest
adding: com/(in = 0) (out= 0)(stored 0%)
adding: com/ibi/(in = 0) (out= 0)(stored 0%)
adding: com/ibi/myoverrides/(in = 0) (out= 0)(stored 0%)
adding: com/ibi/myoverrides/intl/(in = 0) (out= 0)(stored 0%)
adding: com/ibi/myoverrides/intl/templates/(in = 0) (out= 0)(stored 0%)
adding: com/ibi/myoverrides/intl/templates/BIP_PageComponentTemplate.properties(in = 25) (out= 27)(deflated -8%)
adding: com/ibi/myoverrides/intl/templates/BIP_PageComponentTemplate_es.properties(in = 27) (out= 29)(deflated -7%)
H:\>

```

3. Navigate to the location of your JAR file, copy the file and paste it into a folder in the Application Server classpath.
4. Sign in to WebFOCUS as an administrator.
5. Launch the Administration Console.
6. On the Configuration tab, under the Application Settings folder, click *Other*.
7. Select the *Enable Custom Strings* check box, and type the following name in the *Custom Strings* field:

`com.ibi.desired_name`

where:

`desired_name`

Is the name of your JAR file.

The following image shows the completed configuration.



8. Click Save and close the Administration Console.
9. Restart your Application Server.
10. Relaunch WebFOCUS.
11. Create a new collaborative or basic portal.

For more information on creating portals, see the *Business Intelligence Portal* manual.

12. Use your custom symbolic reference to name an element in your portal. In this example, we are using it as a page title. In the page Properties panel, in the Title field, type:

`${value}`

where:

`value`

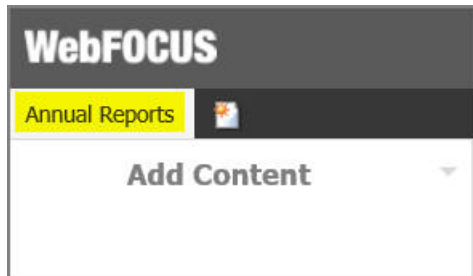
Is the name of your symbolic reference.

An example of a symbolic reference applied in the portal is shown in the following image.



13. Save and close the portal.
14. Run your portal.

The symbolic reference value displays, as shown in the following image.



15. In the portal, on the Menu Bar, click *Language* and select the language you used for your symbolic reference, in this example, select Spanish.

The symbolic reference value changes to its equivalent in the set language, as shown in the following image.



Note: If the Language option is disabled in your WebFOCUS environment, you can enable it in the Administration Console. Under the Dynamic Language Switch settings, select the *Enable Dynamic Language* check box, select the languages that you want to use in your portal from the table, and click Save.

Index

A

- alphanumeric data types [29](#)
- American National Standards Institute (ANSI) code pages [121](#), [124](#)
 - equivalent of server code page [81](#)
 - troubleshooting [107](#), [116](#)
- American Standard Code for Information Interchange (ASCII) code set [18](#), [121](#), [124](#)
- ANSI (American National Standards Institute) code pages [121](#), [124](#)
 - equivalent of server code page [81](#)
 - troubleshooting [107](#), [116](#)
- App Studio [87](#)
- ASCII (American Standard Code for Information Interchange) code set [18](#), [121](#), [124](#)

B

- Big-5 Chinese character set [31](#), [123](#)
- BUSDAYS parameter [141](#)
- BYTVAL function [164](#), [165](#)

C

- case conversion [16](#)
- case translation functions [100](#)
- CDN (Continental Decimal Notation) [129](#)
 - TSGU INFO [111](#)
- CENT-ZERO parameter [131](#)
- cgivars.wfs (client configuration file) [81](#)

- changing NLS configuration settings [116](#)

- CHAR function [166](#)

- character encoding standard [28](#)

- character functions, simplified

 - REPLACE [205](#)

- character functions

 - BYTVAL [164](#), [165](#)

 - CTRAN [167–169](#)

 - DCTRAN [169](#)

 - DSTRIP [211](#), [212](#)

 - LCWORD [198–200](#)

 - LCWORD2 [200](#), [202](#)

 - LCWORD3 [201](#)

 - LOCASE [202](#)

 - STRIP [209–211](#)

 - UPCASE [218](#), [219](#)

- character semantics

 - functions and [100](#)

- character sets [59](#)

 - ASCII [18](#)

 - Double-Byte Character Set (DBCS) [17](#)

 - Single-Byte Character Set (SBCS) [17](#)

- character strings

 - converting case [202](#), [218](#)

 - translating characters [164](#), [167](#)

- Chinese characters [18](#)

- Chinese/Japanese/Korean (CJK) scripts [57](#), [122](#)

- CJK (Chinese/Japanese/Korean) scripts [57](#), [122](#)

- client code pages [16](#), [20](#), [109](#), [111](#), [115](#)

client configuration file (cgivars.wfs) [81](#)

code page 137 [18](#), [124](#)

 default value [79](#)

code page families [123](#)

code page generation file [116](#)

 on Windows, UNIX, and IBM i (cpcodepg.nls)
 [68](#)

code page generation lists [59](#), [116](#)

code page values [59](#), [74](#)

 for client [79](#)

 for IBM i [76](#)

 for UNIX [74](#)

 for Windows [74](#)

code pages [15](#), [17](#), [59](#)

 associating with Master Files [158](#)

 history [121](#)

 identifying [68](#)

 Information Builders [18](#)

 supported [68](#)

code points [20](#)

COLLATION parameter [143](#)

column width in Excel [128](#)

combining data [30](#)

commands [125](#)

 locale [145](#)

 SET CDN [129](#)

 SET CENT-ZERO [131](#)

 SET EXL2KLANG [126](#)

communications configuration file [82](#)

configuration file [65](#), [66](#)

Configuration Wizard [59](#)

configuring a WebFOCUS Client [79](#)

configuring a WebFOCUS Reporting Server [59](#)

configuring NLS (National Language Support) [59](#),
[79](#), [87](#), [116](#)

configuring WebFOCUS Client for NLS (National
Language Support) [79](#)

Continental Decimal Notation (CDN) [129](#)

 TSGU INFO [111](#)

conversion functions, simplified

 CHAR [166](#)

 HEXTYPE [191](#)

cpcodepg.nls file [68](#)

cpxcptbl.nls file [68](#)

creating language resource files [44](#)

CTRAN function [167–169](#)

currency symbols [132](#)

D

data sources [107](#)

 different languages [230](#)

 selecting code page on Windows, UNIX, and
 IBM i [68](#)

 Unicode [28](#), [30](#)

data types [28](#), [29](#), [123](#)

date and time functions

 DATECVT [170](#)

 DATETRAN [172](#)

 HDATE [188](#), [189](#)

- date formats [16](#)
 - international [172](#)
 - setting in sample application [231](#)
- DATE_ORDER parameter [139](#)
- DATE_SEPARATOR parameter [140](#)
- date-time values
 - converting formats [188](#)
- DATECVT function [170](#)
- DATETRAN function [172](#), [180](#)
- DBCS (Double-Byte Character Set) [17](#), [122](#)
 - limitations [119](#)
- DCTRAN function [169](#)
- decimal notation [16](#)
 - setting [113](#), [129](#)
- decoding, comparison and editing functions [100](#)
- default local system encoding [81](#)
- determining NLS values [109](#)
- display issues [117](#)
- displaying leading zeroes [131](#)
- displaying multilingual reports [230](#)
- displaying NLS information for TSGU INFO [111](#)
- Double-Byte Character Set (DBCS) [17](#), [122](#)
 - limitations [119](#)
- double-byte characters [169](#), [211](#)
- DSTRIP function [211](#), [212](#)
- dynamic language switch [36](#)

E

EBCDIC (Extended Binary Coded Decimal Interchange Code) [19](#), [68](#)

- edaprint.log file [109](#)
- editing NLS configuration files [65](#), [66](#)
- encoding [17](#), [28](#)
 - Far Eastern languages [122](#)
- error messages [35](#), [57](#), [106](#)
- Excel column width [128](#)
- Excel language [67](#), [126](#)
- EXCELRELEASE parameter [127](#)
- EXL2K_DEFCOLW parameter [67](#), [128](#)
- EXL2KLANG parameter [67](#), [126](#)
- Extended Binary Coded Decimal Interchange Code (EBCDIC) [19](#), [68](#)
- extended characters [29](#)
- extended currency symbols [132](#)

F

- FILE declaration [146](#)
- filters [114](#)
- Fixed files [95](#)
- Fixed files configuration for Unicode [95](#)
- fixed-width encoding [123](#)
- fonts [104](#)
 - Unicode in PDFs [104](#)
- format conversion functions [100](#)
 - HEXBYT [189](#), [190](#)
 - UFMT [217](#)
- format conversions
 - to characters [189](#)
 - to hexadecimal [217](#)

formatting [15](#)

 currency [132](#)

 dates [16](#)

 numbers [16](#), [129](#), [131](#)

 time [16](#)

functions, LCWORD3 [201](#)

functions

 STRREP [215](#)

G

graphic characters [17](#)

H

HDATE function [188](#), [189](#)

HDAY parameter [141](#)

hexadecimal notation [20](#), [59](#)

HEXBYT function [189](#), [190](#)

HEXTYPE function [191](#)

holiday file [141](#)

I

ibimultilanguage.js file [43](#)

IBIWF_langperm cookie [41](#)

IBIWF_language cookie [41](#)

IBIWF_language parameter [43](#)

 changing values [44](#)

IBM i installation [60](#)

IBM i language and code page values [19](#), [76](#)

identifying supported code page settings [68](#)

ideographs [122](#)

 Japanese (kanji) [18](#), [29](#)

IME (Input Method Editors) [57](#)

Input Method Editors (IME) [57](#)

installing localized versions [34](#)

international date formats [172](#)

interpreting data [16](#)

J

Japanese ideographs (kanji) [18](#)

Japanese kanji [18](#), [29](#)

justification functions [100](#)

K

kanji (Japanese ideographs) [18](#), [29](#)

KKFCUT function [197](#)

L

language values [74](#)

[74](#)

language values

 for IBM i [76](#)

 for Windows and UNIX [74](#)

 passing to URL [42](#)

languages [17](#)

languages and code pages [17](#), [18](#)

 for IBM i [76](#)

 for Windows and UNIX [74](#)

large number formatting [16](#), [129](#)

large numbers [16](#), [129](#)

LCWORD function [198–200](#)
 LCWORD2 function [200](#), [202](#)
 LCWORD3 function [201](#)
 leading zeroes [131](#)
 length and position functions [100](#)
 limitations in WebFOCUS language [119](#)
 LNGPREP utility [153](#)
 local languages [33](#)
 error messages [57](#)
 online Help [57](#)
 local system encoding [81](#)
 locale command [145](#)
 localized software [33](#)
 localized versions [33](#)
 error messages [57](#)
 installing [34](#)
 positional [33](#), [57](#)
 program interface [52](#)
 localizing software [33](#)
 LOCASE function [202](#)

M

Master Files
 associating code pages [158](#)
 encoding schemes [108](#)
 FILE declaration [146](#)
 metadata
 multilingual [153](#)
 Microsoft SQL Server configuration for Unicode [95](#)
 modal encoding [122](#)

monocasing [16](#)
 multi-language code pages [17](#)
 multi-locale applications [221](#)
 multilingual metadata [153](#)
 multilingual reports [230](#)
 multiple scripts [28](#)

N

national characters [18](#)
 history [121](#)
 incorrect display [107](#)
 printing [118](#)
 National Language Support (NLS) values [112](#)
 NLS API
 Unicode support [28](#)
 NLS commands [125](#)
 NLS configuration files
 editing [65](#), [66](#)
 NLS Configuration Wizard [59](#)
 NLSCFG file [66](#)
 nlscfg.err file [65](#), [126](#)
 non-modal encoding [123](#)
 numeric values [20](#)

O

odin.cfg file [83](#)
 online Help [57](#)
 OpenVMS subserver code pages [19](#)
 Oracle character semantics [96](#)
 Oracle configuration for Unicode [96](#)

OS/390 [19](#)

OS/390 OpenEdition code pages [19](#)

P

parameters [158](#)

 CDN [129](#)

 CENT-ZERO [131](#)

 EXL2K_DEFCOLW [128](#)

 EXL2KLANG [126](#)

 IBIWF_language [41](#)

PATTERN function [203](#)

PDF report output

 Unicode fonts [104](#)

PDF reports [160](#)

 displaying NLS characters [160](#)

portals

 localizing [239](#)

positional localized versions [33](#), [57](#)

printing issues [118](#)

procedures

 code pages [108](#)

program interfaces [52](#)

proprietary code page [18](#), [124](#)

 default value [79](#)

PS Reports [160](#)

 displaying NLS characters [160](#)

punctuating large numbers [129](#)

R

reading data [16](#)

README files [107](#)

release notes [107](#)

REPLACE function [205](#)

resetting configuration [107](#)

REVERSE function [207](#)

S

SAP BW configuration for Unicode [97](#)

SAP R/3-ECC configuration for Unicode [97](#)

SBCS (Single-Byte Character Set) [17](#), [122](#)

scripts [28](#)

[17](#)

selecting client code pages [74](#)

selecting server code pages [68](#)

server code pages [20](#)

 for Windows, UNIX, and IBM i [59](#)

server trace file [114](#)

SET CDN command [129](#)

SET CENT-ZERO command [131](#)

SET EXL2KLANG command [126](#)

SET parameters

 BUSDAYS [141](#)

 COLLATION [143](#)

 DATE_ORDER [139](#)

 DATE_SEPARATOR [140](#)

 HDAY [141](#)

 TIME_SEPARATOR [141](#)

 WEEKFIRST [142](#)

setting the version of Excel [127](#)

SFTDEL function [212](#)

SFTINS function [213](#)
 Shift-JIS encoding [29](#), [123](#)
 Single-Byte Character Set (SBCS) [17](#), [122](#)
 single-byte characters [169](#), [211](#)
 software localization [33](#)
 Solaris default locale system encoding [81](#)
 sort order [16](#)
 sort order with Unicode [103](#)
 sorting tables [59](#), [116](#)
 specify multilingual descriptions in a Master File [146](#)
 specifying business days [141](#)
 string manipulation and extraction functions [100](#)
 string replacement [215](#)
 string replacement functions [100](#)
 STRIP function [209–211](#)
 STRREP function [215](#)
 subservers [19](#)
 supported code page file
 on Windows, UNIX, and IBM i (cpxcptbl.nls) [68](#)
 supported code pages [68](#)
 switching between languages [39](#)
 Sybase ASE Adapter
 Unicode support [97](#)
 system configuration [106](#)

T

text encoding [28](#)
 time formats [16](#)

TIME_SEPARATOR parameter [141](#)
 Traditional Chinese [33](#), [123](#)
 scenarios [30](#)
 trans_file attribute [153](#)
 transcoding data [16](#)
 transcoding tables [59](#)
 translated user interface components [34](#)
 troubleshooting display issues [117](#)
 troubleshooting for NLS (National Language Support) [105](#), [108](#)
 troubleshooting printing issues [118](#)
 TSGU INFO command [111](#)

U

UFMT function [217](#)
 Unicode [27](#), [91](#), [123](#)
 access non-FOCUS data sources [93](#)
 Asian characters and [91](#)
 configuring WebFOCUS for UTF-8 [93](#)
 data type support for Relational adapters [98](#)
 Db2 data types [98](#)
 European characters and [91](#)
 Fixed files configuration [95](#)
 fonts for PDFs [104](#)
 manipulating characters [100](#)
 Microsoft SQL data types [98](#)
 Microsoft SQL Server configuration [95](#)
 Oracle configuration [96](#)
 Oracle data types [98](#)
 PDF output [104](#)

Unicode [27](#), [91](#), [123](#)

 reformatting characters [100](#)

 SAP BW configuration [97](#)

 SAP R/3-ECC configuration [97](#)

 sort order and [103](#)

 Sybase ASE Adapter [97](#)

 Sybase ASE data types [98](#)

universal character encoding standard [27](#), [91](#)

UNIX configuration [59](#), [79](#)

UNIX language and code page values [18](#), [74](#)

unrelated scripts [28](#)

UPCASE function [218](#), [219](#)

user interface components [34](#)

UTF (Unicode Transformation Format) [91](#)

UTF-16 Unicode [28](#), [123](#)

UTF-8 Unicode

 configuration scenarios [30](#)

V

VM subserver code pages [19](#)

W

WebFOCUS Administration Console [44](#), [79](#)

WebFOCUS Client

 configuring for NLS [79](#)

WebFOCUS Reporting Server Console [59](#)

 displaying server code page [109](#)

 displaying server trace [114](#)

 running TSGU [111](#)

WebFOCUS Reporting Server

 configuring for NLS [59](#)

 localized versions [35](#)

WEEKFIRST parameter [142](#)

Windows configuration [59](#), [79](#), [87](#)

Windows language and code page values [18](#), [74](#)

Z

z/OS [19](#)



Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at Sarah_Buccellato@ibi.com.

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at Frances_Gambino@ibi.com.

WebFOCUS



National Language Support
Release 8.2 Version 02