

z/OS



Support for Unicode: Using Unicode Services

z/OS



Support for Unicode: Using Unicode Services

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 485.

Eleventh Edition, September 2008

This is a major revision of SA22-7649-09.

This edition applies to Version 1 Release 10 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405

FAX (Other Countries):

Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrdfs@us.ibm.com

World Wide Web: <http://www.ibm.com/systems/z/os/zos/webqs.html>

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2001, 2008. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Tables	ix
About this document	xi
Who should use this document	xi
How this document is organized	xi
Overview of contents	xi
Syntax diagrams	xii
Where to find more information	xii
Information updates on the web	xii
Summary of changes	xiii
Chapter 1. Introduction	1
The Unicode Standard	1
z/OS support for Unicode	1
The Unicode environment	2
The infrastructure	2
The conversion services	3
National language support	6
Chapter 2. Setting up the system	7
Prerequisites	7
Creating the Unicode environment	7
Updating parmlib members	7
MVS Message Service	7
Using the pre-built DB2 conversion image	8
Chapter 3. Creating a Unicode Services environment	9
Creating a conversion image	9
Step a: Select the conversions	10
Step b: Specify control statements	11
Image generator	15
Step c: Invoke the image generator	15
Step d: Use the image generator listing	16
Specifying the type of conversion	19
Calculating the storage needed for a conversion image	21
Changing the conversion environment	23
Chapter 4. Character conversion	25
Calling the character conversion services	25
Restrictions for the calling environment	29
Using the C interface	29
Mapping of parameters in C	29
31-bit mapping	30
64-bit mapping	31
Using the HLASM interface	33
Mapping of parameters for AMODE (31)	34
Description of parameters in area CUNBCPRM	36
Mapping of parameters for AMODE (64)	44
Description of parameters in area CUN4BCPR	47
Handling a target buffer overflow	55

Character conversion service and the new B technique	57
Sample programs	57
Chapter 5. Case conversion	59
Calling the case conversion services	59
Restrictions for the calling environment	60
Using the C interface	60
Mapping of parameters in C	61
31-bit mapping	61
64-bit mapping	62
Using the HLASM interface	62
Mapping of parameters for AMODE (31)	63
Description of parameters in area CUNBAPRM	65
Mapping of parameters for AMODE (64)	70
Description of parameters in area CUN4BAPR	71
Sample programs	76
Chapter 6. Normalization.	77
Calling the normalization service	77
Handling a work buffer overflow.	78
Restrictions for the calling environment	78
Using the C interface	78
Mapping of parameters in C	79
31-bit mapping	79
64-bit mapping	80
Using the HLASM interface	80
Mapping of parameters for AMODE (31)	81
Description of parameters in area CUNBNPRM	83
Mapping of parameters for AMODE (64)	85
Description of parameters in area CUN4BNPR	87
Sample programs	89
Chapter 7. Collation	91
Calling the collation service	92
Restrictions for the calling environment	97
Using the C interface	97
Mapping of parameters in C	101
31-bit mapping	102
64-bit mapping	103
Mapping of constants in C	105
Using the HLASM interface	107
Mapping of parameters for AMODE (31)	110
Description of parameters in area CUNBOPRM	113
Mapping of constants for AMODE (31).	128
Mapping of parameters for AMODE (64)	130
Description of parameters in area CUN4BOPR.	133
Mapping of constants for AMODE (64).	147
Sort key vector format.	149
Work buffer length considerations	150
Target buffer length considerations	151
Sample programs	152
Chapter 8. Bidi transformation	155
Calling Bidi transformation service	155
Using the C interface	156
Mapping of parameters in C	156

31-bit mapping	156
64-bit mapping	157
Using the HLASM interface	157
Mapping of parameters for AMODE (31)	158
Description of parameters in area CUNBBPRM	159
Mapping of parameters for AMODE (64)	161
Description of parameters in area CUN4BBPR	162
Character conversion service and the new B technique	163
Chapter 9. Stringprep conversion	165
Calling the stringprep services	165
Using the C interface	166
Mapping of parameters in C	166
31-bit mapping	166
64-bit mapping	167
Using the HLASM interface	168
Mapping of parameters for AMODE (31)	169
Description of parameters in area CUNBPPRM	170
Mapping of parameters for AMODE (64)	173
Description of parameters in area CUN4BPPR	174
Sample programs	176
Chapter 10. Conversion information service	177
Calling the conversion information service	177
Restrictions for the calling environment	177
Using the C interface	178
Mapping of parameters in C	178
31-bit mapping	178
64-bit mapping	181
Using the HLASM interface	183
Mapping of parameters for AMODE (31)	188
Description of parameters in area CUNBIPRM	193
Mapping of parameters for AMODE (64)	201
Description of parameters in area CUN4BIPR	207
Sample programs	215
Appendix A. MBCS conversions	217
Internal handling of MBCS conversions	217
MBCS CCSID decomposition	219
MBCS CCSIDs compatible with iconv	222
C-variant MBCS CCSIDs compatible with iconv()	222
Appendix B. Conversion Tables Supplied with z/OS Unicode	223
Description of CCSIDs	223
EBCDIC Conversion Table	248
ASCII Conversion Table	326
Unicode Conversion Table	419
Case Conversion tables	442
Normalization tables	444
Collation tables	446
Stringprep tables.	448
Appendix C. Defining CCSIDs and conversion tables	449
Defining a CCSID in the knowledge base.	449
Specifying the entries	449
Assembling the source data set	451

Creating a conversion table	452
Building a human-readable character map	452
Converting the character map to binary format.	454
Modifying job CUNJIUTL.	454
Appendix D. Unicode Return and Reason Codes	457
Return code meanings	457
Image generator for z/OS support for Unicode – return codes	467
Appendix E. System control offsets to callable services	469
Examples for 31-bit callers	469
List of offsets for 31-bit services	469
Examples for 64-bit callers	469
List of offsets for 64-bit services	469
Appendix F. Locales support	471
Appendix G. Case service and locale valid names	479
Appendix H. Accessibility	483
Using assistive technologies	483
Keyboard navigation of the user interface.	483
z/OS information.	483
Notices	485
Policy for unsupported hardware	486
Trademarks.	486
Glossary of terms and abbreviations	489
Index	495

Figures

1. Conversion of MBCS data to Unicode characters	218

Tables

1.	Encoding schemes	3
2.	CCSID conversions types of z/OS support for Unicode	4
3.	Main storage needed for conversions of type SBCS and DBCS.	21
4.	Main storage needed for conversions of type MBCS.	21
5.	Restrictions while calling the character conversion services	29
6.	Mapping of parameters in HLASM for character conversion AMODE (31)	34
7.	Mapping of parameters in HLASM for character conversion AMODE (64)	44
8.	Minimum and maximum character widths of the different encoding schemes	55
9.	Restrictions while calling the case conversion services	60
10.	Mapping of parameters in HLASM for case conversion AMODE (31)	63
11.	Case Conversion Service supported locales - AMODE(31)	67
12.	Mapping of parameters in HLASM for case conversion AMODE (64)	70
13.	Case Conversion Service supported locales - AMODE(64)	73
14.	Unicode version table	77
15.	Restrictions while calling the normalization service	78
16.	Mapping of parameters in HLASM for normalization AMODE (31)	82
17.	Mapping of parameters in HLASM for normalization AMODE (64)	86
18.	Restrictions for the calling environment.	97
19.	Mapping of parameters in HLASM for collation AMODE (31)	110
20.	Collation mask sub fields descriptions.	117
21.	Equivalencies between short path and long path locale settings	121
22.	Collation keywords descriptions	122
23.	Valid values for collation keywords	124
24.	Collation rule symbols	125
25.	Collation syntax rules.	125
26.	Mapping of parameters in HLASM for collation AMODE (64)	130
27.	Collation mask sub fields descriptions.	137
28.	Equivalencies between short path and long path local settings	141
29.	Collation keywords descriptions	141
30.	Valid values for collation keywords	143
31.	Collation rule symbols	144
32.	Collation syntax rules.	145
33.	Collation level weight length	150
34.	Size of the work buffers for UTF-16BE Code Points	151
35.	Recommended target buffer lengths for collation.	151
36.	Size of the target buffers for UTF-16BE Code Points	151
37.	Target Buffer Formula	152
38.	The AMODE and API (C/C++ or HLASM) in combination with long or short path settings	152
39.	Mapping of parameters in HLASM for Bidi AMODE (31)	158
40.	Mapping of parameters in HLASM for Bidi AMODE (64)	161
41.	Mapping of parameters in HLASM for stringprep AMODE (31).	169
42.	Mapping of parameters in HLASM for stringprep AMODE (64).	173
43.	Restrictions while calling the conversion information service services	177
44.	Mapping of parameters in HLASM for conversion information service AMODE (31)	188
45.	Mapping of parameters in HLASM for conversion information service AMODE (64)	202
46.	EBCDIC Conversion Table	288
47.	ASCII Conversion Table	326
48.	Unicode Conversion Table	419
49.	Case conversion service based on the Unicode Standard 3.0.1.	442
50.	Case conversion service based on the Unicode Standard 3.2.0.	443
51.	Case conversion service based on the Unicode Standard 4.0.1.	443
52.	Case conversion service based on the Unicode Standard 4.1.0.	443
53.	Case conversion service based on the Unicode Standard 5.0.0.	444

54.	Normalization service based on the Unicode Standard 3.0.1.	444
55.	Normalization service based on the Unicode Standard 3.2.0.	445
56.	Normalization service based on the Unicode Standard 4.0.1.	445
57.	Normalization service based on the Unicode Standard 4.1.0.	446
58.	Collation service based on the Unicode Standard 3.0.1.	446
59.	Collation service based on the Unicode Standard 4.0.0.	447
60.	Collation service based on the Unicode Standard 4.1.0.	447
61.	Profiles provided for stringprep service	448
62.	Naming conventions of conversion tables	455
63.	Classification of return codes	457
64.	Return and reason codes from Unicode Services	457
65.	Offsets for 31-bit callers.	469
66.	Offsets for 64-bit callers.	469
67.	Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit)	471
68.	Case service and locale valid names	479

About this document

This document provides guidance for using z/OS® support for Unicode Services.

Who should use this document

This document is intended for system programmers and system administrators who want to know how to set up and use the Unicode Services environment.

How this document is organized

Following is an overview of the contents of this document and some additional relevant information.

Overview of contents

This document contains the following information:

- Chapter 1, “Introduction,” on page 1 is an overview of what the Unicode Standard is and what Unicode support on the z/OS platform is.
- Chapter 2, “Setting up the system,” on page 7 includes the installation of z/OS support for Unicode Services.
- Chapter 3, “Creating a Unicode Services environment,” on page 9 describes the tasks before activating the Unicode Services conversion environment: creating a conversion image, calculating the storage needed for the conversion image, and initializing the conversion environment.
- Chapter 4, “Character conversion,” on page 25 gives instructions on how to use the character conversion services.
- Chapter 5, “Case conversion,” on page 59 gives instructions on how to use the case conversion services.
- Chapter 6, “Normalization,” on page 77 gives instructions on how to use the normalization services.
- Chapter 7, “Collation,” on page 91 gives instructions on how to use the collation services.
- Chapter 8, “Bidi transformation,” on page 155 describes the programming required for the Bidi transformation service.
- Chapter 9, “Stringprep conversion,” on page 165 describes the programming required for the stringprep conversion services.
- Appendix A, “MBCS conversions,” on page 217 describes how MBCS conversions are handled internally.
- Appendix B, “Conversion Tables Supplied with z/OS Unicode,” on page 223 shows all tables IBM® provides for conversions.
- Appendix C, “Defining CCSIDs and conversion tables,” on page 449 shows how you can define your own knowledge base entries, coded character set identifiers (CCSIDs), and conversion tables.
- Appendix D, “Unicode Return and Reason Codes,” on page 457 lists the Unicode Services return and reason codes.
- Appendix E, “System control offsets to callable services,” on page 469 describes the system control offsets that can be used as an alternative to linking or link-editing the service stub.
- Appendix F, “Locales support,” on page 471 lists the locales supported in the data set SYS1.SCUNLOCL.

About this document

- “Glossary of terms and abbreviations” on page 489 explains the terminology used in this document.

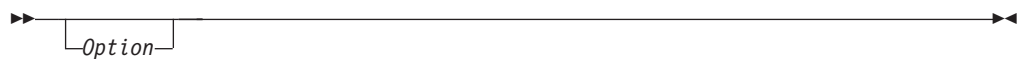
Syntax diagrams

This document uses railroad syntax diagrams to illustrate how to use commands. This is how you read a syntax diagram:

A command or keyword that you must enter (a required command) is displayed like this:



An optional keyword is shown below the line, like this:



A default is shown over the line, like this:



An item that can be repeated is shown like this:



Where to find more information

Where necessary, this document references information in other documents using shortened versions of the document title. For complete titles and order numbers of the books for all products that are part of z/OS, see *z/OS Information Roadmap*.

The following document contains additional information you might need when using z/OS Unicode Services.

- Character Data Representation Architecture Reference .

Additional information on the Unicode® Consortium can be found at the Web site <http://www.unicode.org/>.

Information updates on the web

For the latest information updates that have been provided in PTF cover letters and Documentation APARs for z/OS, see the online document at:

<http://www.s390.ibm.com:80/bookmgr-cgi/bookmgr.cmd/BOOKS/ZIDOCMST/CCONTENTS>

This document is updated weekly and lists documentation changes before they are incorporated into z/OS publications.

Summary of changes

Summary of changes for SA22-7649-10 z/OS Version 1 Release 10

The document contains information previously presented in *z/OS Support for Unicode: Using Conversion Services*, SA22-7649-09, which supports z/OS Version 1 Release 9.

New information:

- You can use the conversion information service to obtain information about details of one specific coded character set identifier (CCSID) or two CCSIDs. See Chapter 10, “Conversion information service,” on page 177.
- The Character conversion contains new output bit flags CUNBCPRM ETF3E_Behavior_ Status and CUNBCPRM ETF3E_Behavior (31 and 64 bit respectively). See Chapter 4, “Character conversion,” on page 25.
- The Collation contains the new collation versions UCA400R1 and UCA410. See Chapter 7, “Collation,” on page 91.

Changed information:

- Appendix Conversion Tables Supplied with z/OS Unicode has been updated. See Appendix B, “Conversion Tables Supplied with z/OS Unicode,” on page 223.

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes for SA22-7649-09 z/OS Version 1 Release 9 as updated April 2008

The document contains information previously presented in *z/OS Support for Unicode: Using Conversion Services*, SA22-7649-08, which supports z/OS Version 1 Release 9.

Changed information

- The Character conversion chapter has been updated. See Chapter 4, “Character conversion,” on page 25.
- The Case conversion chapter has been updated. See Chapter 5, “Case conversion,” on page 59.
- Appendix Conversion Tables Supplied with z/OS Unicode has been updated. See Appendix B, “Conversion Tables Supplied with z/OS Unicode,” on page 223.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

**Summary of changes
for SA22-7649-08
z/OS Version 1 Release 9**

The document contains information previously presented in *z/OS Support for Unicode: Using Conversion Services*, SA22-7649-07, which supports z/OS Version 1 Release 8.

New information

- A new appendix has been added to list all the valid locales names for Case Service. See Appendix G, “Case service and locale valid names,” on page 479.

Changed information

- The Character conversion chapter has been updated. See Chapter 4, “Character conversion,” on page 25.
- The Case conversion chapter has been updated. See Chapter 5, “Case conversion,” on page 59.
- The Collation® chapter has been updated. See Chapter 7, “Collation,” on page 91.
- Appendix Conversion Tables Supplied with z/OS Unicode has been updated. See Appendix B, “Conversion Tables Supplied with z/OS Unicode,” on page 223.
- Appendix Unicode Return and Reason Codes has been updated. See Appendix D, “Unicode Return and Reason Codes,” on page 457.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

**Summary of changes
for SA22-7649-07
z/OS Version 1 Release 8**

The document contains information previously presented in *z/OS Support for Unicode: Using Conversion Services*, SA22-7649-06, which supports z/OS Version 1 Release 7.

New information

- The Unicode conversion environment now includes four image generation steps to create a conversion image.
- The Unicode services now introduce a new "B" technique for Bidi in conjunction with the character conversion service.
- The Unicode collation service now supports customization. Collation service might behave according to certain collation rules specified using a Locale or by some User Collation Rules (UCR).
- New tables have been added for Normalization and Collation service in support of Unicode upgrade to 4.0.
- A new appendix has been added to include the locales supported for collation service. See Appendix F, “Locales support,” on page 471.

Changed information

New chapters have been created for:

- Bidi
- Stringprep

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Chapter 1. Introduction

z/OS V1R10 support for Unicode is based on Version 4.1.0 of the Unicode Standard, although lower versions are supported by some services, review each individual service to see the Unicode versions supported.

For further information on the standard and the Unicode Consortium, go to the Web site at <http://www.unicode.org/>.

The Unicode Standard

This text is from the Unicode Consortium Web site.

The Unicode Standard is the universal character encoding standard used for representation of text for computer processing. It is fully compatible with the second edition of International Standard ISO/IEC 10646-1:2000, and contains all the same characters and encoding points as ISO/IEC 10646. The Unicode Standard also provides additional information about the characters and their use. Any implementation that is conformant to Unicode is also conformant to ISO/IEC 10646.

Unicode provides a consistent way of encoding multilingual plain text and brings order to a chaotic state of affairs that has made it difficult to exchange text files internationally. Computer users who deal with multilingual text -- business people, linguists, researchers, scientists, and others -- will find that the Unicode Standard greatly simplifies their work. Mathematicians and technicians, who regularly use mathematical symbols and other technical characters, will also find the Unicode Standard valuable.

The design of Unicode is based on the simplicity and consistency of ASCII, but goes far beyond ASCII's limited ability to encode only the Latin alphabet. The Unicode Standard provides the capacity to encode all of the characters used for the written languages of the world. It uses a default 16-bit encoding that provides code points for more than 65,000 characters. To keep character coding simple and efficient, the Unicode Standard assigns each character a unique numeric value and name.

While 65,000 characters are sufficient for encoding most of the many thousands of characters used in major languages of the world, the Unicode standard and ISO 10646 provide an extension mechanism called UTF-16 that allows for encoding as many as a million more characters, without use of complex modes or escape codes. This is sufficient for all known character encoding requirements, including full coverage of all historic scripts of the world.

z/OS support for Unicode

z/OS support for Unicode includes character conversion, case conversion, normalization, collation, stringprep conversion and bidirectional transformation.

- Within character conversion, characters are converted from one coded character set identifier (CCSID) to another.

Introduction

- Case conversion allows conversion to upper or lower case based on the files UnicodeData.txt and SpecialCasing.txt provided by the Unicode Consortium.
- Normalization allows the decomposition or composition of Unicode encoding input to any of four normalization forms.
- Collation allows for culturally correct comparisons between two Unicode strings. It can also provide a sort key for one or two input Unicode strings for later use in binary comparisons.
- The stringprep conversion service prepares a string of Unicode text in order to increase the likelihood that string input and string comparison work in ways that make sense for typical users.
- Bidirectional transformation defines a minimal set of directional formatting codes to control the ordering of characters when rendered. This allows exact control of the display ordering for legible interchange and also ensures that plain text used for simple items like filenames or labels can always be correctly ordered for display.

z/OS support for Unicode consists of two main components:

- The infrastructure, which provides the Unicode environment
- The conversion services, which use the Unicode environment

The Unicode environment

The Unicode conversion environment is available for use at all times. Whether it was created by specifying a UNI=xx statement in the system parameters for IPL or an empty environment was activated by default, the Unicode environment is ready for use after IPL has completed. A non-empty Unicode environment is the common-storage data spaces loaded with tables and other data needed during Unicode services call.

Starting in z/OS V1R7, the Unicode environment can be dynamically updated. Callers can invoke Unicode services and the tables required for respective service will be loaded as they are referenced. The operator can also update the environment with the SETUNI command to add, delete, or replace tables. See Chapter 3, “Creating a Unicode Services environment,” on page 9.

The infrastructure

The purpose of the infrastructure is to establish and maintain the Unicode environment. The infrastructure supplies all parts to customize and initialize the Unicode services. All necessary resources are allocated and tables are loaded into storage as defined in the image. Changes to the Unicode environment can be performed with the **SET UNI** and **SETUNI** command.

The user can specify the required conversions and control information in a text file. A program called the image generator processes the text file and generates an Unicode image containing all specified tables. The image can be activated with the **SET UNI** command, the **SETUNI** command, or with the next IPL.

The infrastructure also includes a **DISPLAY UNI** command to show the current status of the Unicode environment or to determine which tables are already loaded into storage.

IBM provides libraries of conversion tables, normalization tables, and collation tables.

z/OS support for Unicode allows customers to add up to ten different user-defined conversion tables for conversions between the same pair of CCSIDs by using the reserved numeric (0-9) techniques. Customers can also define new CCSIDs for their own purposes. These CCSIDs must not conflict with the predefined CCSIDs shipped with z/OS support for Unicode.

See Appendix C, “Defining CCSIDs and conversion tables,” on page 449 for the definition of new CCSIDs.

The conversion services

The conversion services make up the other major part of z/OS support for Unicode. They consist of a variety of conversion types and permit character conversion as well as case conversion for which the appropriate conversion tables were provided by the infrastructure.

Encoding schemes

A basic feature of a CCSID is its encoding scheme, which is uniquely identified by the encoding scheme identifier (ESID). In this document, the following names are used for the encoding schemes:

Table 1. Encoding schemes

ESID Hex	Interpretation
1100	EBCDIC, SBCS
2100	IBM-PC Data, SBCS
3100	IBM-PC Display, SBCS
4100	ISO 8, SBCS
4105	ISO 8 (ASCII code), SBCS
4155	ISO 8 Presentation (ASCII code), SBCS
5100	ISO 7 (ASCII code), SBCS
5150	ISO 7 Presentation (ASCII code), SBCS
6100	EBCDIC Presentation, SBCS
8100	8 bit, SBCS, used with a 7-bit code page
1200	EBCDIC, DBCS
2200	IBM-PC Data, DBCS
3200	IBM-PC Display, DBCS
5200	ISO 7 (ASCII code), DBCS
7200	Unicode, including UTF-16 to allow for support of surrogates
8200	Unicode display
9200	8 bit, DBCS, used with a 7-bit code page
1301	EBCDIC, Mixed single-byte and double-byte, using SO/SI code extension method
2300	IBM-PC Data, Mixed single-byte and double-byte, with implicit code extension
2305	IBM PC Data, Mixed single byte and double-byte, SBCS
3300	IBM-PC Display, Mixed single-byte and double-byte, with implicit code extension
4403	IBM EUC
5404	ISO 2022 TCP/IP using ESC sequences

Table 1. Encoding schemes (continued)

ESID Hex	Interpretation
5409	ISO 2022 TCP/IP using SO/SI
540A	ISO 2022 TCP/IP using SO, SI, SS2, SS3
5700	ISO 7 Triple-Byte Code Set
720B	Unicode UTF-32, Little endian order. No code extension is allowed.
7500	Unicode UTF-32, BigEndian order. No code extension is allowed.
7807	UTF-8, UCS-2 transform
2900	PC Data, fixed 4-byte
2A00	PC Data, mixed single-byte, double-byte, four-byte

For other encoding schemes, refer to Character Data Representation Architecture Reference .

Code pages with a pure single-byte or pure double-byte encoding (SBCS, DBCS, and UCS-2) are called simple code pages. Code pages that consist of two or more sub code pages (PC MBCS, EUC MBCS, EBCDIC MBCS, and ISO2022 MBCS) are called mixed code pages.

UCS-2 may be encoded in big endian or in little endian format. z/OS support for Unicode only handles the big endian form.

z/OS support for Unicode does not handle surrogate pairs except for conversions from and to UTF-8. z/OS support for Unicode interprets CCSID 1200 (UCS2) as the latest available version of the Unicode Standard.

UTF-16 might be encoded in big endian or little endian format. The default of z/OS support for Unicode is big endian format, an order in which the "big end" is stored first. CCSID 1202 is defined to be UTF-16 little endian, an order in which the "little end" is stored first.

Character conversion

z/OS support for Unicode provides direct conversion between character streams that are encoded with CCSIDs listed in Appendix B, "Conversion Tables Supplied with z/OS Unicode," on page 223. Character conversion is also called conversion between specified CCSIDs. The following CCSID conversions types are supported for direct conversions:

Table 2. CCSID conversions types of z/OS support for Unicode

SBCS	<=>	SBCS, DBCS
DBCS	<=>	SBCS, DBCS
PC MBCS	<=>	DBCS
EUC MBCS	<=>	DBCS
EBCDIC MBCS	<=>	DBCS
ISO2022 MBCS	<=>	DBCS
UTF-8	<=>	UCS-2
QBCS	<=>	DBCS

For an explanation of the terms, refer to "Glossary of terms and abbreviations" on page 489.

For character conversion, the conversion services are called using a stub routine named **CUNLCNV** for AMODE (31), or **CUN4LCNV** for AMODE (64). z/OS support for Unicode must be called in primary mode.

Besides the direct conversions, there are indirect conversions where any CCSID can be converted into another by using the intermediate CCSID 1200. The indirect conversion with CCSID 1200 is automatically used by z/OS support for Unicode if there is no table available for direct conversions between FROM-CCSID and TO-CCSID, but there are tables available from FROM-CCSID to CCSID 1200 and from CCSID 1200 to TO-CCSID.

The conversion of MBCS characters also uses several steps to complete the conversion. This is called a composite conversion. An MBCS input data stream is decomposed into SBCS and DBCS parts. The conversion services automatically select an SBCS table for the SBCS data and a DBCS table for the DBCS data. There are no MBCS tables provided by z/OS support for Unicode. You can find a detailed description of the internal handling in Appendix A, “MBCS conversions,” on page 217. An example and an illustration is included.

Case conversion

z/OS support for Unicode provides case conversions that allow users to convert Unicode characters to their upper case equivalent or their lower case equivalent. For more details about the case mappings, refer to the tables provided by the Unicode Consortium.

For case conversion, the conversion services are called using a stub routine named **CUNLASE** for AMODE (31) or **CUN4LASE** for AMODE (64).

Normalization

z/OS support for Unicode provides support that allows the normalization (decomposition or composition) of Unicode characters to one of the normalization forms. For a detailed explanation of normalization, including specific information about the normalization forms, refer to the Technical Report #15 provided by the Unicode Consortium (<http://www.unicode.org/unicode/reports/tr15/>).

The normalization service is called using a stub routine named **CUNLNORM** for AMODE (31) or **CUN4LNOR** for AMODE (64).

Collation

z/OS Support for Unicode provides the Collation Service to make a culturally correct binary comparison between two Unicode strings. It can also generate a sort key, which can later be used by the caller to do binary comparisons between strings. For a detailed explanation of the Unicode collation process, please refer to the Unicode Consortium Technical Report #10 at: <http://www.unicode.org/unicode/reports/tr10>.

The collation service is called using a stub routine named **CUNLOCOL** for AMODE (31) or **CUN4LCOL** for AMODE (64).

Bidi

z/OS support for Unicode provides Bidirectional Text support that allows users to order character strings according to their display properties. For a detailed explanation of Bidi, refer to the Technical Report #9 provided by the Unicode Consortium (<http://www.unicode.org/unicode/reports/tr9/>).

The bidi transformation service is called using a stub routine named **CUNLBIDI** for AMODE (31) or **CUN4LBID** for AMODE (64).

Introduction

Stringprep

z/OS support for Unicode provides String preparation for internationalized string useful for some internet protocols. This feature is based on RFC 3454.

The String preparation service is called using a stub routine named **CUNLSTRP** for AMODE (31) or **CUN4LSTP** for AMODE (64).

Conversion information

z/OS support for Unicode provides Conversion information for obtaining information about details of one specific coded character set identifier (CCSID) or two CCSIDs. The conversion information service is used separately or is used before the z/OS Unicode character conversion service.

The conversion information service is called using a stub routine named **CUNLINFO** for AMODE (31) and **CUN4LINP** for AMODE (64).

National language support

z/OS support for Unicode provides a Japanese translation of its messages. For details see “MVS Message Service” on page 7.

Chapter 2. Setting up the system

This chapter describes how you set up the your system to use Unicode Services. z/OS support for Unicode is integrated in the z/OS system.

Prerequisites

For information about z/OS hardware and software prerequisites, see *z/OS Planning for Installation*.

The z/OS V1R8 data sets that are required for Unicode Services are:

- SYS1.SCUNTB, which contains all of the Unicode Services tables shipped from IBM.
- SYS1.LPALIB and SYS1.LINKLIB, which contain Unicode Services program modules.
- SYS1.SCUNIMG, which contains the prebuilt DB2® HC2 image shipped from IBM. This data set must be in an APF-authorized library in the LNKST concatenation. Be aware that lower level versions of this library will not work with z/OS V1R8.
- SYS1.SCUNLOCL, which contains all the LOCALES of Collation services.

Creating the Unicode environment

Prior to establishing the conversion environment, you must do the following:

- Update the required parmlib members.
- Determine if you need to use the MVS™ Message Service (MMS).
- Investigate the use of the pre-built conversion image for DB2.

Updating parmlib members

The parmlib members that you must update are CUNUNlxx and IEASYSxx.

CUNUNlxx

CUNUNI contains information that the system needs to activate, replace, or delete Unicode conversion environments. The conversion environment is set up to create a conversion image that is loaded into storage. The conversion image will contain the conversion tables that define the data conversions allowed between CCSIDs. For information about creating this parmlib member, see Chapter 3, “Creating a Unicode Services environment,” on page 9 and CUNUNlxx in *z/OS MVS Initialization and Tuning Reference*.

IEASYSxx

IEASYSxx contains system parameters. The UNI parameter of IEASYSxx specifies the CUNUNlxx parmlib member for your conversion environment. See IEASYSxx in *z/OS MVS Initialization and Tuning Reference*.

MVS Message Service

z/OS support for Unicode provides for Japanese translation of its messages. The MVS Message Service (MMS) is used to convert English Unicode messages to the equivalent Japanese messages and to modify the date and time formats of messages. *z/OS MVS Planning: Operations* provides detailed information about setting up and using MMS. A brief summary appears here.

Setting up the system

- Allocate storage for a VSAM linear data set that will be used to contain the run-time message files. Unicode Services provides sample job CUNJIMS1 for this allocation. See how to allocate the data set in *z/OS MVS Planning: Operations*.
- Compile the message skeleton to create the input to the run-time message files. Unicode Services provides an English message skeleton, CUNIIENU, and a Japanese message skeleton, CUNIIJPN, in \$CUN_MSG_DS\$. Unicode provides sample job CUNJIMS2 for this compilation. See complete directions for compiling the message files in *z/OS MVS Planning: Operations*.
- Update the appropriate parmlib members needed for MMS. To control information about each language on your system, use the MMSLSTxx parmlib member. To control configuration data for displaying date and time formats, use the CNLcccxx parmlib member. To specify the MMSLSTxx member to be activated, use the INIT statement of CONSOLxx. See the relevant parmlib information in *z/OS MVS Planning: Operations*.

Using the pre-built DB2 conversion image

z/OS Unicode Services support provides a way of automatically loading a pre-built conversion image into the Unicode environment. This conversion image provides the most commonly used codepage conversion tables needed to satisfy the DB2 codepage conversion needs. This feature eliminates the need to pre-set specific settings to enable the DB2-required Unicode Services conversion environment.

This DB2-required Unicode conversion environment, represented by the pre-built conversion image, will be established at IPL- time if the following two conditions are successfully met:

1. No UNI=xx parameter is provided in IEASYSxx. This results in the system establishing an empty conversion environment.
2. Unicode Services are successfully called by DB2 during its initialization process. This results in establishing the pre-built image in the Unicode environment. To validate the existence of the pre-built image, issue the DISPLAY UNI,STORAGE command. If the pre-built image was loaded, the command results will show approximately 10000 pages of storage loaded for the active image.

Note: The size of a pre-built conversion image can change from release to release.

If a failure occurs and both conditions are not successfully met, the pre-built image will not be loaded. You must then load a conversion image that will satisfy DB2 requests into the system.

Member CUNSISM6 of SYS1.SAMPLIB contains the conversion statements needed to build the DB2 pre-built image.

Chapter 3. Creating a Unicode Services environment

A Unicode Services environment is set when a conversion or service request is loaded. A conversion image consists of the conversion tables and control blocks used by the conversion services. This chapter describes how to:

- Create a conversion image.
- Calculate the amount of storage needed for a conversion image.
- Handle error conditions that occur within the conversion environment.
- Change the conversion environment.
- Set the Unicode Services environment.

Creating a conversion image

A conversion image is a single entity that holds all necessary information to support one callable services configuration. A conversion image can be loaded into the system during IPL or by issuing the SET UNI or SETUNI command. The layout of a conversion image is:

- Image header
- Control information
- Unicode Services tables
- Image trailer

Prior to z/OS V1R7, the Unicode Services environment had to be established with all required tables loaded into storage for use by the conversion services before a caller could successfully invoke a service. If the appropriate table was not loaded, a new image containing the table had to be built and loaded into storage with either an IPL or a SET UNI command.

Starting in z/OS V1R7, the Unicode Services environment can be dynamically updated when a conversion service is requested. If the appropriate table needed for the service is not already loaded into storage, Unicode Services will load the table without requiring an IPL or disrupting the caller's request.

The new Unicode Services interfaces provided starting in V1R7 are an expanded CUNUNlxx parmlib member and a SETUNI operator command that accomplish the same function as the parmlib member. With either of these interfaces you can:

- Add, replace, or delete tables in a conversion image, specifying the FROM-CCSID, the TO-CCSID, and optionally, the techniques required.
- Add, replace, or delete case conversion tables.
- Add, replace, or delete normalization tables.
- Add, replace, or delete collation tables.
- Add, replace, or delete Stringprep profiles.
- Add an image, without requiring that it be in the parmlib concatenation.

Multiple images can be kept in data sets. Using the SET UNI or SETUNI command they can be used to complement the Unicode Services environment by merging them into the image (duplicated conversion tables or dropped-only deltas are merged into the environment).

Unicode Services uses the following when creating the conversion image:

Creating a Unicode Services environment

1. Knowledge base (supplied by IBM): describe the CCSIDs that are supported. The knowledge base is contained in module CUNMIKBS and found in SYS1.LINKLIB.
2. Conversion tables (supplied by IBM): located in SYS1.SCUNTB. Unicode Services transforms the conversion tables into an internal format and stores them in the conversion image.
3. Input statements (either from CUNUNlxx or from the SETUNI command): describe which of the conversions are to be included in the conversion image. The CCSIDs used in each input statement must be defined in the knowledge base. For each pair of CCSIDs that describes a conversion, one or more conversion tables must exist (depending whether this is a simple or composite conversion).

You may also have user-defined CCSIDs and conversion tables. For details see Appendix C, “Defining CCSIDs and conversion tables,” on page 449.

The image generator creates the following output:

- A conversion image. The conversion image is built according to the specification in the SYSIN DD data set. Each required character conversion is described by a CONVERSION control statement. Case conversion can be requested using the CASE control statement, normalization with the NORMALIZE control statement, and collation with the COLLATE control statement. The generated image is stored in the data set specified in the //SYSIMG DD statement.
- A listing on the //SYSPRINT DD statement that shows the processed steps and error messages if applicable. For a detailed description of the image generator listing, see “Image generator” on page 15.
- A return code.

To create a conversion image, follow these steps (a – d):

Step a: Select the conversions

There are four types of conversion:

1. Character conversion between two different CCSIDs.
2. Case conversion for Unicode characters.
3. Normalizing of a Unicode string.
4. Collation, for culturally correct comparison between two Unicode strings.

For character conversions, each CCSID pair between which you want to be able to convert using the conversion services has to be identified. However, there are different techniques to convert between two CCSIDs and you can specify your preferred technique(s):

(R) Roundtrip conversion

Roundtrip conversions between two CCSIDs assure that all characters making the “roundtrip” arrive as they were originally.

(E) Enforced Subset conversion

Enforced Subset conversions map only those characters from one CCSID to another that have a corresponding character in the second CCSID. All other characters are replaced by a substitution character.

(C) Customized conversion

Customized conversions use conversion tables that have been created to address some special requirements.

(L) Language Environment-Behavior conversion

Language Environment-Behavior conversions use tables that map characters like the `iconv()` function of the Language Environment® Runtime library does. These conversions differ from others primarily in their mapping of the EBCDIC newline (NL) character to ASCII and Unicode linefeed (LF).

(M) Modified Language Environment-Behavior conversion

Modified Language Environment-Behavior conversions use tables that map characters like the `iconv()` function of the Language Environment Runtime library does for converters ending with "C" (for example IBM-932C).

(0-9) User-defined conversions

User-defined conversions are supported. See Appendix C, "Defining CCSIDs and conversion tables," on page 449.

For case conversion you can have the following conversion modes:

- **NORMAL casing:**
This means that one character is mapped to its upper/lower case using a one-to-one relationship as described in the file `UnicodeData.txt`. Characters that cannot be mapped are copied to the output stream unchanged. Note also that locale specific casing is not supported with mode NORMAL. NORMAL is the preferred mode for converting English text.
- **SPECIAL casing:**
In addition to NORMAL casing, locale independent special casing as listed in the file `SpecialCasing.txt` is performed. This can be unconditional special casing (for example, 'German Small Letter Sharp s' = X'00DF' uppercases to 2 characters of 'Capital Letter S' =X'00530053') or conditional special casing (for example, 'Greek Capital Letter Sigma'=X'03A3' lowercases to either 'Greek Small Sigma'=X'03C3' when within a word or to 'Greek Small Final Sigma'=X'03C2' when it is the last character of a word).
- **LOCALE dependent casing:**
In addition to SPECIAL casing, locale dependent special casing as listed in the file `SpecialCasing.txt` is performed (for example, 'Capital Letter I' =X'0049' lowercases to 'Small Letter i'=X'0069' when caller's language is NOT turkish, but lowercases to 'Small Letter Dotless i'=X'0131" when caller's language is Turkish CUNBCPRM_Locale='tr...').

Note: Note that user-defined case conversions are not supported.

For normalization and collation services, no special mode is required. See "Normalization conversion" on page 21 and "Collation conversion" on page 21.

Step b: Specify control statements

There are four different control statements that can be specified in the `//SYSIN DD` statement of job CUNJIUTL:

- **CONVERSION** (for character conversion)
- **CASE** (for case conversion)
- **NORMALIZE** (for normalization)
- **COLLATE** (for collation)

Control statement **CONVERSION**

Purpose

Creating a Unicode Services environment

Each CONVERSION control statement defines exactly one conversion that should be generated in the conversion image. We call this a 'top-level conversion'. Duplicate CONVERSION statements are ignored. It is possible that the image generator uses more than 1 table to reflect the CONVERSION statement. This might be because MBCS CCSID is involved or a particular conversion table needed was not found. In the case of MBCS involvement, the system implements a composite conversion with a set of sub-level conversions according to its knowledge base. In the case of missing conversion tables, an indirect conversion – using CCSID 1200 as the intermediate CCSID – is generated.

In general, a direct conversion is supported when:

- converting between any combination of SBCS and DBCS
- converting between MBCS and DBCS
- converting between UTF-8 and UCS-2

All other conversions will always be indirect conversions.

Format

►► CONVERSION *from-ccsid* , *to-ccsid* . *technique-search-order* ; ►

technique-search-order:**technique-character:**

Diagram illustrating a stack structure. The stack is represented by a vertical container with cells. The cells are labeled from top to bottom: *blank*, R, E, C, L, M, and 0-9. A horizontal line extends from the right side of the *blank* cell.

Parameters

From-ccsid

The value from-ccsid specifies the FROM-CCSID of the requested conversion. The FROM-CCSID is the CCSID you are converting from.

To-ccsid

The value to-ccsid specifies the TO-CCSID of the requested conversion. The TO-CCSID is the CCSID you are converting to.

Technique-search-order

There may be multiple conversion tables available for converting one CCSID to another. A technique-search-order can be used to specify which table should be used. It consists of up to 8 technique-characters. If you specify more than one technique character, the image generator will try to find a matching table for the

leftmost technique—character in the sequence of the technique-search-order. If not found, the search continues with the second one and so on. A blank character terminates the search. Especially for mixed conversion, it is advisable to use more than one technique-character as one of the sub-conversions might exist only in round-trip mode and one only in enforced-subset. In this case, a technique-search-order of 'RE' or 'ER' would be required. Technique-search-order is optional. If not specified, RECLM is used.

Some of the tables were fixed for use with MBCS conversions. These tables have the internal technique 'X' and override tables with technique 'C' or 'R'. Fixed tables for technique 'L' have the internal technique 'Y'. The image generator will first check existence of 'X' tables before searching for tables with technique 'C' or 'R'. It will also first check the existence of 'Y' tables before searching for tables with technique 'L'. The output of the image generator lists which table actually was selected. Technique characters 'X' and 'Y' cannot be specified by customers. When calling the service, there are two ways to request the technique search:

1. Using the default technique search order (RECLM) or without any (a blank) in the CONVERSION, the field CUNBCPRM_Technique of the parameter area can contain RECLM or a blank.
2. The same character sequence as specified in the technique-search-order of the CONVERSION control statement must be specified in the CUNBCPRM_Technique field of the parameter area.

Technique-character

Possible values for technique-character are:

- R: Roundtrip
- E: Enforced Subset
- C: Customized Subset
- L: Language Environment Behavior
- M: Modified Language Environment Behavior
- 0 – 9: User-defined conversions

Some special considerations about CCSID 1200: If CCSID 1200 is specified, the CCSID of the most recent UCS-2 version is substituted and all technique-characters are tested. Then the second recent UCS-2 version is substituted and so on. The supported UCS-2 CCSIDs are:

- 21680 (UCS-2 V3.1)
- 17584 (UCS-2 V3)
- 13488 (UCS-2 V2)

Here are some examples of valid CONVERSION statements:

```
CONVERSION 850,037;      /* technique-search-order omitted, use RECLM */
CONVERSION 850,037,;     /* duplicate, this line will be ignored */
CONVERSION 850,037,R;    /* will use Roundtrip */
CONVERSION 933,13488,RE; /* will search fixed 'Xotables,Roundtrip, then */
                        /* Enforced Subset */
```

Control statement CASE

Purpose

The CASE control statement selects the case conversions that should be generated in the conversion image.

Creating a Unicode Services environment

►►—CASE—*mode*—◄◄

mode:



Parameters

mode

specifies the case conversion mode to be supported. The following modes are supported:

- NORMAL - basic casing, preferred mode for English text
- SPECIAL - includes normal casing, adds locale independent special casing
- LOCALE - includes special casing, adds locale dependent special casing

Examples

Here is an example of a valid CASE statement:

```
CASE NORMAL; /* normal casing requested */
CASE NORMAL; /* Duplicate CASE statements are ignored */
CASE LOCALE; /* locale dependent special casing requested */
```

Control statement NORMALIZE

Purpose

The NORMALIZE control statement loads the normalization tables in the conversion image.

►►—NORMALIZE—◄◄

Parameters

None

Examples

Here is an example of a valid NORMALIZE statement:

```
NORMALIZE; /* normalization requested */
NORMALIZE; /* Duplicate NORMALIZE statements are ignored */
```

Control statement COLLATE

Purpose

The COLLATE control statement loads the collation tables in the conversion image.

►►—COLLATE—◄◄

Parameters

None

Examples

Here is an example of a valid COLLATE statement:

```
COLLATE; /* collation requested */
COLLATE; /* Duplicate COLLATE statements are ignored */
```

Image generator

Once you have selected the conversions and specified the control statements, you can continue creating the conversion image by invoking the image generator and using the image generator listing.

Step c: Invoke the image generator

Invoke the image generator for z/OS support for Unicode. Member CUNJIUTL in library SCUNJCL contains the JCL to invoke the image generator:

```
// $JOBPREF$ JOB ($ACCOUNT$), '$USER$', NOTIFY=$NOTIFY$,
//   MSGCLASS=$MC$, MSGLEVEL=$ML$, TIME=$TI$, CLASS=$CL$,
//   REGION=$REGION0M$
// *****
// *
// * IMAGE GENERATOR
// *
// *****
// CUNMIUTL EXEC PGM=CUNMIUTL
// SYSPRINT DD SYSOUT=*
// TABIN DD DSN=$CUN_TBL_DS$, DISP=SHR
// SYSIMG DD DSN=$CUN_IMAGE_DS$(CUNIMG00), DISP=SHR
// SYSIN DD *

/*****
* INPUT STATEMENTS FOR THE IMAGE GENERATOR *
*****/

NORMALIZE;          /* ENABLE NORMALIZATION */
COLLATE;             /* ENABLE COLATION */
CASE NORMAL;         /* ENABLE TOUPPER AND TOLOWER */
CASE LOCALE;         /* ENABLE LOCALE */
CASE SPECIAL;        /* ENABLE SPECIAL */
CONVERSION 1047,850; /* EBCDIC -> ASCII */
CONVERSION 850,1047; /* ASCII -> EBCDIC */
```

Step d: Use the image generator listing

The sample JCL from step (c) produces the following listing on the //SYSPRINT DD:

```
Source Listing ----+----1----+----2----+----3----+----4----+----5----+----6----+
1
2      /*****
3      * INPUT STATEMENTS FOR THE IMAGE GENERATOR *
4      *****/
5
6 NORMALIZE;                /* ENABLE NORMALIZATION          */
7 COLLATE;                  /* ENABLE COLATION              */
8 CASE NORMAL;              /* ENABLE TOUPPER AND TOLOWER   */
9 CASE LOCALE;              /* ENABLE LOCALE                */
10 CASE SPECIAL;            /* ENABLE SPECIAL               */
11 CONVERSION 1047,850;      /* EBCDIC -> ASCII              */
12 CONVERSION 850,1047;      /* ASCII -> EBCDIC              */
13
Statement Report --+---1---+---2---+---3---+---4---+---5---+---6---+
1 CONVERSION 1047,850;;
   /* 01047-00850-R using CUNRM0EB */
2 CONVERSION 850,1047;;
   /* 00850-01047-R using CUNREBM0 */
3 CASE NORMAL;
   /* to-upper normal using CUNANUUP */
   /* to-lower normal using CUNANULO */
4 CASE LOCALE;
   /* to-upper locale using CUNASCUP */
   /* special casing table using CUNASCAS */
   /* category table using CUNASCLT */
   /* to-lower locale using CUNASCLO */
   /* special casing table using CUNASCAS */
   /* category table using CUNASCLT */
5 CASE SPECIAL;
   /* to-upper special using CUNASUUP */
   /* special casing table using CUNASCAS */
   /* category table using CUNASCLT */
   /* to-lower special using CUNASULO */
   /* special casing table using CUNASCAS */
   /* category table using CUNASCLT */
6 NORMALIZE;
   /* canonical decomposition table..... using CUNNCDTB */
   /* canonical decomposition stop table.... using CUNNC DST */
   /* compatibility decomposition table..... using CUNNKDTB */
   /* compatibility decomposition stop table. using CUNNK DST */
   /* composition table..... using CUNNCOMT */
   /* composition stop table..... using CUNNCOST */
   /* canonical class table..... using CUNNCACT */
   /* canonical class non zero table ..... using CUNNCCNZ */
7 COLLATE;
   /* CE Main Table..... using CUNOBACE */
   /* Expansions Index Table..... using CUNOEXIN */
   /* Expansion Elements Table..... using CUNOEXDA */
   /* Contraction Index Table..... using CUNOTIDX */
   /* Contraction Elements Table..... using CUNOCODA */
   /* Main Index Table..... using CUNOMIDX */
   /* Rearrangement Values - Thai and Lao..... using CUNOTHLA */
   /* Fast Canonical Decomposition Stop Table... using CUNOFCD */
   /* Fast Compatibility Decomposition Stop Tbl.. using CUNOFKD */
   /* Fast Composition Stop Table..... using CUNOFCO */
CUN1014I INPUT READ          13 RECORDS
CUN1015I STATEMENTS PROCESSED      7
CUN1016I STATEMENTS FLAGGED        0
CUN1017I GENERATED IMAGE SIZE    522 PAGES
CUN1002I PROCESSING ENDED. HIGHEST RETURN CODE WAS 0
```

The listing can be divided into four sections:

1. The identification section. This section shows the product version and when the job was started.

2. The source listing. This section repeats the data from //SYSIN DD exactly as entered.
3. The statement report. This section shows the recognized statements and how they were resolved.
4. The statistic section. This section gives an overview of the complete process.

The following descriptions explain how the listing can be used to manage the generated images.

The identification section

If you have already generated a lot of images and keep them in data sets, it might be of interest to match an image generator listing with an existing image. For this reason there is a readable time stamp in the first record of the image. This time stamp matches the time stamp on message CUN1001I.

```
CUN1000I Z/OS SUPPORT FOR UNICODE VERSION V1R6
CUN1001I PROCESSING STARTED ON 01/29/2004 AT 12:11:09
```

The source listing

Especially when concatenated data sets are used on the //SYSIN DD statement, it is important to check which control statements were provided in the input stream. The source listing shows exactly what was read from //SYSIN DD and the number that is assigned to each input record.

The statement report

In the statement report you can see what the image generator has interpreted from the provided input. All comments, blanks, and line breaks have been removed. Each recognized statement is printed in a normalized form and a statement number is assigned. Comments are inserted after the statement to explain what was generated by the system.

```
...
Statement Report --+---1---+---2---+---3---+---4---+---5---+---6---+
1  CONVERSION 933,1200,RE;
    /* 00933-01200-RE */
    /* 00833-01200-R      using CUNRDIPG */
    /* 00834-01200-R      using CUNRDMPG */
...
```

The left hand side in the comment shows a hierarchy of the top-level and sub-level conversions. The right hand side shows the name of the tables used.

The statistics sections

The most important information from the statistic section is the return code. If the return code is 0, processing was successful from the technical point of view. You should always check the statement report carefully to ensure the generated image contains the necessary tables and correct CCSIDs.

Error situations: The following paragraphs show how the listing can be used in error situations:

1. Environmental errors:

Before processing starts all the required resources are checked and allocated. When errors occur in that phase no source listing and no statement report are generated. The identification and statistic sections are printed. No image is generated. A listing with an environmental error might look like this:

Creating a Unicode Services environment

```
CUN1000I Z/OS SUPPORT FOR UNICODE VERSION V1R6
CUN1001I PROCESSING STARTED ON 01/29/2004 AT 14:13:14

CUN1007E ERROR OCCURRED OBTAINING TEMPORARY WORK STORAGE RC=00000004

CUN1014I INPUT READ                0 RECORDS
CUN1015I STATEMENTS PROCESSED      0
CUN1016I STATEMENTS FLAGGED        0
CUN1017I GENERATED IMAGE SIZE      0 PAGES
CUN1002I PROCESSING ENDED. HIGHEST RETURN CODE WAS 12
```

2. Syntactical errors:

Once the initialization phase has successfully been executed the input stream is read from //SYSIN DD and the source listing is produced. The input stream then is parsed for syntactical errors. The values of the parameters are not checked at this point. Syntactical errors are for instance:

- unrecognized statement keywords
- missing/excessive parameters
- missing/excessive commas or semicolons

The statement report is not printed. No image is generated. A listing with a syntactical error might look like this:

```
CUN1000I Z/OS SUPPORT FOR UNICODE VERSION V1R6
CUN1001I PROCESSING STARTED ON 01/29/2004 AT 15:16:17

Source Listing ----+----1----+----2----+----3----+----4----+----5----+----6---+
1
2  /*****
3    * INPUT STATEMENTS FOR THE IMAGE GENERATOR *
4    *****/
5
6  NORMALIZE;                /* ENABLE NORMALIZATION      */
7  COLLATE;                  /* ENABLE COLATION          */
8  CASE NORMAL;              /* ENABLE TOUPPER AND TOLOWER */
9  CASE LOCALE;              /* ENABLE LOCALE            */
10 CASE SPECIAL;             /* ENABLE SPECIAL           */
11 CONVERSION 1047;          /* EBCDIC -> ASCII          */
12 CONVERSION 850,1047;      /* ASCII -> EBCDIC          */
13
CUN4005E MANDATORY PARAMETER(S) MISSING FROM STATEMENT
'CONVERSION' IN LINE 11.
A MINIMUM OF TWO PARAMETERS IS REQUIRED

CUN1014I INPUT READ                13 RECORDS
CUN1015I STATEMENTS PROCESSED      0
CUN1016I STATEMENTS FLAGGED        0
CUN1017I GENERATED IMAGE SIZE      1 PAGES
CUN1002I PROCESSING ENDED. HIGHEST RETURN CODE WAS 8
```

3. Semantical errors:

When the syntax of a statement is correct the specified parameters are checked for reasonable values. Semantical errors are for instance:

- CCSIDs out of range
- invalid *technique-characters*
- invalid case conversion modes
- conversion table not found

The statement is printed in the statement report followed by the error messages issued. No image is generated. A listing with a semantical error might look like this:

```

CUN1000I Z/OS SUPPORT FOR UNICODE VERSION V1R6
CUN1001I PROCESSING STARTED ON 01/29/2004 AT 15:23:14

Source Listing -----1-----2-----3-----4-----5-----6---+
1
2      /*****
3      * INPUT STATEMENTS FOR THE IMAGE GENERATOR *
4      *****/
5
6      NORMALIZE;                /* ENABLE NORMALIZATION      */
7      COLLATE;                  /* ENABLE COLATION          */
8      CASE NORMAL;              /* ENABLE TOUPPER AND TOLOWER */
9      CASE LOCALE;              /* ENABLE LOCALE            */
10     CASE SPECIAL;             /* ENABLE SPECIAL           */
11     CONVERSION 1047,85000;      /* EBCDIC -> ASCII          */
12     CONVERSION 850,1047;        /* ASCII -> EBCDIC          */
13

Statement Report --1-----2-----3-----4-----5-----6---+
1     CONVERSION 1047,85000;;
CUN1023E ERROR DURING CCSID VALIDATION. INVALID CCSID '85000'
2     CONVERSION 850,1047;;
      /* 00850-01047-R using CUNREBM0      */
3     CASE NORMAL;
      /* to-upper normal using CUNANUUP    */
      /* to-lower normal using CUNANULO    */
4     CASE LOCALE;
      /* to-upper locale using CUNASCUP     */
      /* special casing table using CUNASCAS */
      /* category table using CUNASCLT      */
      ....
      /* Fast Canonical Decomposition Stop Table.... using CUNOFCD */
      /* Fast Compatibility Decomposition Stop Tbl.. using CUNOFKD */
      /* Fast Composition Stop Table..... using CUNOFCD */
CUN1014I INPUT READ              13 RECORDS
CUN1015I STATEMENTS PROCESSED    7
CUN1016I STATEMENTS FLAGGED     7
CUN1017I GENERATED IMAGE SIZE   1 PAGES
CUN1002I PROCESSING ENDED. HIGHEST RETURN CODE WAS 8

```

After generating the conversion image, copy it to SYS1.PARMLIB or any other data set in the logical parmlib concatenation.

After completing the steps a to d, continue with “Calculating the storage needed for a conversion image” on page 21.

Specifying the type of conversion

Use statements in the CUNUNlxx parmlib member or on the SETUNI command to specify the type of conversions required by your installation.

Character conversion

For character conversion, use the ADD (or REPLACE or DELETE) FROM(yyyyy) TO(yyyyy) statement. Duplicate statements are ignored. It is possible that Unicode Services uses more than one table to reflect the CONVERSION statement. This might be because an MBCS CCSID is involved or a particular conversion table needed was not found. In the case of MBCS involvement, the system implements a composite conversion with a set of sub-level conversions according to its knowledge base. In the case of missing conversion tables, an indirect conversion – using CCSID 1200 as the intermediate CCSID – is generated.

In general, a direct conversion is supported when:

- Converting between any combination of SBCS and DBCS
- Converting between MBCS and DBCS

Creating a Unicode Services environment

- Converting between UTF-8 and UCS-2.

All other conversions will always be indirect conversions.

The parameters that can be specified for character conversion are:

FROM-CCSID

Specifies the FROM-CCSID of the requested conversion. This is the CCSID you are converting from.

TO-CCSID

Specifies the TO-CCSID of the requested conversion. This is the CCSID you are converting to.

TECHNIQUE

Specifies the technique to be used in the conversion.

Possible values for *technique-character* are:

- R: Roundtrip
- E: Enforced Subset
- C: Customized Subset
- L: Language Environment Behavior
- M: Modified Language Environment Behavior
- 0 – 9: User-defined conversions

Some special considerations for CCSID 1200: If CCSID 1200 is specified, the CCSID of the most recent UCS-2 version is substituted and all *technique-characters* are tested. Then the second recent UCS-2 version is substituted and so on. The supported UCS-2 CCSIDs are:

- 21680 (Unicode 4.0)
- 17584 (Unicode 3.0)
- 13488 (Unicode 2.0)

Understanding how Unicode Services loads conversion tables: When you specify one or more techniques for a particular character conversion, Unicode Services loads all appropriate tables for the requested conversion into the image. If you do not specify any technique, then Unicode Services loads all available tables for the requested conversion into the image. For example, if you specify FROM-CCSID=1208, TO-CCSID=875, and technique=ERC, then Unicode Services will load tables for 1208-875-E, 1208-875-R, or 1208-875-C. Additional tables will be loaded if no technique is specified on the request. At run time, if a request for a conversion does not include a technique, Unicode uses a default search order, R E C L M and 0 - 9, to assign a conversion table to the request.

Composite conversions (those that require different techniques in the intermediate steps) require the use of Sub CCSIDs to perform a conversion. Unicode Services determines those techniques that will be used and stores them into the image. If you do not specify any technique for a composite conversion, then Unicode Services loads only the first table found for each Sub CCSID.

Case conversion

For case conversion, use the ADD (or REPLACE or DELETE) CASE statement.

Optional parameters that can be specified on the CASE statement define the conversion mode to be supported. You can specify one or more of the conversion mode parameters, but duplicates will be ignored.

NORMAL

Specifies basic casing, preferred mode for English text.

SPECIAL

Specifies normal casing and adds locale independent special casing.

LOCALE

Specifies special casing and adds locale-dependent special casing.

Normalization conversion

For normalization conversion, use the ADD (or REPLACE or DELETE) NORMALIZATION statement. The normalization versions that can be specified are:

- UNI301
- UNI320
- UNI401
- UNI410

Collation conversion

For collation conversion, use the ADD (or REPLACE or DELETE) COLLATION statement. The collation versions that can be specified are:

- UCA301
- UCA400R1
- UCA410

Calculating the storage needed for a conversion image

Following are the steps you need to perform to calculate the main storage needed for a conversion image.

Estimating the size of an image based on planned conversions

To estimate the size of main memory an image would require depending on its set of conversions, use the following rule of thumb:

- For conversion tables, use the size in the following tables:

Table 3. Main storage needed for conversions of type SBCS and DBCS

conversion type	size of storage
SBCS→SBCS	0.25 KB
SBCS→DBCS	0.50 KB
DBCS→SBCS	64.00 KB
DBCS→DBCS	128.00 KB
QBCS→DBCS	128.00 KB
DBCS→QBCS	162.00 KB

The sizing in the following table is based on the assumption that the MBCS CCSID consists of one SBCS and one DBCS codepage.

Table 4. Main storage needed for conversions of type MBCS

conversion type	size of storage
MBCS→SBCS direct	64 KB
MBCS→SBCS via 1200	192 KB
MBCS→DBCS direct	128 KB
MBCS→DBCS via 1200	256 KB

Creating a Unicode Services environment

Table 4. Main storage needed for conversions of type MBCS (continued)

conversion type	size of storage
MBCS→MBCS via 1200	320 KB
GB18030 MBCS→DBCS	257 KB
DBCS→GB18030 MBCS	291 KB

If a MBCS CCSID is composed differently, break it into its sub-CCSIDs and calculate the size for each part separately, according to Table 3.

- For any type of case conversion add 256 KB for the main casing tables. As soon as any of the types SPECIAL or LOCALE casing are used, add another 58 KB for additional tables.
- For the case conversion statement also add 0.25 KB for control structures. For indirect and composite conversions add 0.25 KB for the control structures of each sub-level conversion.
- For the normalization statement add 565 KB, which is the total size of the tables needed for normalization as shown in Appendix C, “Defining CCSIDs and conversion tables,” on page 449.
- For Collation tables, refer to Appendix C, “Defining CCSIDs and conversion tables,” on page 449.
- For any conversion involving a table where the source is Unicode Double Byte, an additional validation table (to validate the malformed characters) is loaded. This validation table is 64 KB in size.

After the image is generated, look for message CUN1017I. It shows exactly the number of pages the image requires in main storage.

Note: Due to DASD configuration, the image stored on DASD occupies about 1.13 times the size.

Since z/OS V1R7, the algorithm to build an image has been enhanced. Unicode Services now loads all available tables for the requested conversion when building an image. For example, prior to z/OS V1R7, if you specified FROM-CCSID=0037, TO-CCSID=0256, and Technique=ER, and both tables were provided by the system, Unicode Services loaded only the first tables specified in the Technique Search Order, namely the table for 0037-0256-E. Starting in z/OS V1R7, Unicode Services now loads the tables for both 0037-0256-E and 0037-0256-R. Therefore, if an existing image is rebuilt since z/OS 1.7, the size of the image will grow because of the additional tables added to the image. You must therefore calculate the amount of storage needed for each conversion when building an image depending on the number of techniques specified on the conversion request and also depending on tables placed in the image. This may also require a reevaluation of the amount of real storage to load the image.

Determining the size of an image from an existing member

The size of an image stored in a data set is different from when it is loaded in main storage. You can calculate the amount of main storage required after loading as follows:

- Load the image in the VIEW ENTRY PANEL from ISPF.
- Go to the last line.
- Multiply the last line number by 71 and divide it by 4096.
- Ignore the decimal places.
- The result is the number of pages needed for that image.

Determining the size of the active image

To get information on the size of the active image loaded to the conversion environment, use the DISPLAY UNI command. Enter

```
DISPLAY UNI,STORAGE
```

and check the command output on section STORAGE. The output looks like:

```
CUN3000I 09.39.07 UNI DISPLAY 476
      STORAGE: ACTIVE      566 PAGES
              LIMIT      123456 PAGES
```

The size of the active image in pages is found after the ACTIVE parameter. In this example 566 pages are used.

Determining the value for the REALSTORAGE parameter

The REALSTORAGE parameter in the CUNUNlxx parmlib member was introduced to protect the z/OS system against main storage shortage, caused by loading a conversion image which exceeds the amount of available real storage. To control the real storage usage, the loading of a new conversion image or individual service request will be rejected when the REALSTORAGE available is less than the amount of storage needed for the complete environment.

The minimum value for the REALSTORAGE parameter is two pages and contains only the necessary information to hold future environment update requests.

The REALSTORAGE value is used as a 'high water mark' and represents the amount of real storage you are willing to dedicate for use by z/OS support for Unicode.

Note: Due to the dynamic capabilities of Unicode Services, the dynamic load of images can require up to the REALSTORAGE reported by the CUNMITUL Image generator's (CUN1017I message) plus 22 pages. The difference in the space required depends on the size of the image.

Managing a conversion handle that is not valid

Each SET UNI command invalidates all conversion handles because the tables they point to may have changed. Each call to a conversion service checks before conversion whether the used handle is valid.

If a conversion handle is not valid, the caller can specify with a flag whether the conversion has to be terminated or retried with a new valid conversion handle. Specify "Terminate with error", for example, if the conversion has to use exactly one version of the conversion table. Specify "Get new handle and continue" if the caller does not need a special version of the conversion table.

Changing the conversion environment

Starting in z/OS V1R7, you can change the conversion environment by either manipulating specific tables within your current environment or by re-IPLing with a new CUNUNlxx parmlib member. The Unicode Services environment can also be dynamically updated by a caller's request for a conversion table that is currently not available in storage.

- Use the SETUNI command to add, replace, or delete conversion tables within your environment. The changes take effect immediately. You can verify the changes with the DISPLAY UNI command. See *z/OS MVS System Commands*.

Creating a Unicode Services environment

- Use the SET UNI=xx command to specify a new CUNUNlxx parmlib member. Once loaded, you can make subsequent changes to the contents of the image with the SETUNI command. See *z/OS MVS Initialization and Tuning Reference*.
- Changes to the current environment also can occur dynamically when a conversion request is received and the environment doesn't support the requested service. Unicode Services loads the tables required for conversion as they are referenced.

Note: New Collation and Normalization features are NOT supported as part of the off-line tool CUNMIUTL for build images purposes. All new collation features will be exploited as part of the Unicode Dynamic Capabilities. See Chapter 7, "Collation," on page 91 and Chapter 6, "Normalization," on page 77 for more information.

Chapter 4. Character conversion

Programming Interface information

This chapter describes the programming required for the character conversion services.

The character conversion is also referred to as conversion between specified CCSIDs. The character conversion services are called using a stub routine named **CUNLCNV** for AMODE (31) and **CUN4LCNV** for AMODE (64). The routine converts a string of text characters between the specified code pages given as CCSIDs. It works for all character conversions that have a UCCE (Unicode character conversion control entry), which is built during the image generation process and is available after IPL. It may be changed by a SET UNI or SETUNI command.

The character conversion environment can be activated by specifying the CONVERSION control statement in the input data set for the image generator (job CUNJIUTL). For detailed information see “Creating a conversion image” on page 9 and “Character conversion” on page 19. The character conversion environment can also be dynamically activated when a conversion request is performed and the requested conversion has not been previously loaded.

The CCSID is defined as a 32-bit binary integer where numbers below X'DFFF' represent standard CCSIDs. (See Character Data Representation Architecture Reference). The range from X'E000' to X'FFFF' can be used for user-defined CCSIDs. Values from X'F000' to X'FFFF' are reserved for special purposes.

All data is expected to be in big endian encoding. Conversion from and to little endian encoding has to be done outside the z/OS support for Unicode. See “Glossary of terms and abbreviations” on page 489 for an explanation of the *endian* format.

Instead of the CCSIDs, a handle to the UCCE can be given as input. This is always possible after the first call because the handle of the UCCE that was used is returned. This helps to speed up the conversion because the code needed to locate the conversion table has to be executed only in the first call.

Note: All indirect conversion services require a work buffer to be provided by the caller of the services. Caller allocation of the work buffer eliminates the need for the services themselves to be concerned with memory management (and cleanup on failure). To hold at least one Unicode character the length of the work buffer in bytes must be at least 2. For optimal performance it should be not less than two times the number of characters in the source string.

Calling the character conversion services

This is a general description of how the character conversion services have to be called and what problems can occur.

The character conversion services convert data between any CCSID types as described in Table 2 on page 4. The recommended DDA size for the character conversion services is 8K, set in the CUNBCPRM_DDA_BUF_LEN and CUN4BCPR_DDA_BUF_LEN fields in the parameter list.

The 31-bit caller of the conversion services must provide the following fields in the parameter area:

- Source buffer pointer, ALET, and length
- Target buffer pointer, ALET, and length (see Note 2)
- FROM-CCSID (or conversion handle in subsequent calls)
- TO-CCSID (or conversion handle in subsequent calls)
- Work buffer pointer, ALET, and length (see Note 2)
- Dynamic data area pointer (DDA), ALET, and length
- Flags

Notes:

1. A dynamic data area (DDA) must always be specified. The required length is defined by constant CUNBCPRM_DDA_REQ for AMODE (31). See Interface Definition File CUNBCIDF.
2. To take advantage of a performance improvement, specifically for EBCDIC <=> UTF8 conversions, the application developer can provide a larger work buffer and/or target buffer. The work buffer and target buffer must be three times the size of the source buffer. Expressed mathematically:

Wrk Buffer Len >= 3* Src Buffer Len AND
Targ Buffer Len >= 3* Src Buffer Len

The 64-bit caller of the conversion services must provide the following fields in the parameter area:

- Source buffer (64 bit pointer), ALET (4 byte), and length (8 byte)
- Target buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte) (see Note 2)
- FROM-CCSID (or conversion handle in subsequent calls)
- TO-CCSID (or conversion handle in subsequent calls)
- Work buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte) (see Note 2)
- Dynamic data area pointer (DDA), ALET, and length (see Note 1)
- Flags

Notes:

1. A dynamic data area (DDA) must always be specified. The required length is defined by constant CUN4BCPR_DDA_REQ for AMODE (64). See Interface Definition File CUN4BCID
2. To take advantage of a performance improvement, specifically for EBCDIC <=> UTF8 conversions, the application developer can provide a larger work buffer and target buffer. The work buffer and target buffer must be three times the size of the source buffer. Expressed mathematically:

Wrk Buffer Len >= 3* Src Buffer Len AND
Targ Buffer Len >= 3* Src Buffer Len

From the caller's perspective, conversions are always done with a single call to the conversion services. However internally, conversions between

- a mixed code page and anything other than simple code pages
- UTF-8 and anything other than UCS-2

are done in two steps. Two step conversions require that a work buffer be supplied by the caller. For coding simplicity, a caller may choose to always supply a work

buffer (which will go unused for single-step conversions). Alternatively, if the caller knows that a particular conversion is "single-step", the work buffer need not be supplied.

The dynamic data area (DDA) is needed to hold all the variables needed internally by the conversion service. The size of the DDA required depends on the type of conversion being done (source and target CCSIDs). The current minimum DDA size is 2000 bytes, and the maximum is 4096 bytes, but the latter is subject to change in future releases as additional conversions are added.

There are two options for specifying DDA size:

1. The maximum DDA size required for any type of conversion may always be specified. This value (currently 4096) is defined by the constant `CUNBCPRM_DDA_Req`, and is subject to change in future releases, requiring a program recompile.
2. The minimum DDA size (defined by constant `CUNBCPRM_DDA_Buf_Min`) may initially be specified for all conversions. If this DDA size is not large enough to support the type of conversion specified by `Src_CCSID` and `Trg_CCSID`, the conversion services will return with a return code of `"CUN_RC_USER_ERR"` and reason code of `"CUN_RS_DDA_BUF_SMALL"`, and will also return the DDA size required for the specified conversion in field `"UCCE_DDA_BUF_LEN"` of the UCCE handle.

Regardless of how the caller determines the DDA size, IBM recommends that the caller also provide code to recognize and react (by allocating a larger DDA buffer and recalling the service) to a `"CUN_RS_DDA_BUF_SMALL"` error.

When the service returns, it updates the source buffer and target buffer pointers, and lengths. Thus the caller can see how many bytes were converted and how much of the target buffer is filled up. Return codes and reason codes notify when a target buffer overflow was detected or other error occurred. Recommendations for the work buffer and target buffer sizes are listed in "Handling a target buffer overflow" on page 55.

The source buffer may contain characters that have no equivalent in the TO-CCSID or may contain the substitution character in the FROM-CCSID. The user of the conversion services specifies the action to take on detection of such a character by the value of the input parameter bit `'CUNBCPRM_Sub_Action'`. Depending on this input bit the conversion service either terminates conversion with reason code `CUN_RS_SUB_ACT_TERM` or it inserts the conversion table's substitution character into the target buffer, sets bit `CUNBCPRM_Substitution` in the parameter list, and continues conversion with the next character in the source buffer.

The source buffer may also contain byte-strings that do not represent a character in the source code page. These characters are referred to as "malformed characters" and cannot be converted to a valid target codepoint. If the `CUNBCPRM_Flag1` parameter bit `CUNBCPRM_Sub_Action` specifies "substitute", and `CUNBCPRM_Mal_Action` specifies "terminate", then the conversion will terminate with `RC=4` and `RS=0C` when a malformed character is encountered. But if `CUNBCPRM_Mal_Action` specifies "substitute", the malformed character will be substituted. The only exception will be when converting from UTF-8 and GB-18030 code pages, where all characters that have no equivalence in the TO-CCSID will be treated as malformed characters. This will result in `RC=4` and `RS=0C`, when `CUNBCPRM_Flag1` parameter bit `CUNBCPRM_Sub_Action` is set to "terminate",

and the CUNBCPRM_Mal_Action parameter bit is ignored. If both CUNBCPRM_Sub_Action and CUNBCPRM_Mal_Action specify "substitute", normal substitution will occur.

The source code page (FROM-CCSID), target code page (TO-CCSID), and technique-search-order are given initially. A call with those specified always returns a conversion handle which – for the services – is a fast path to the conversion table and its properties. In subsequent calls, it is recommended that the caller provides the conversion handle. If a caller wants to request the conversion handle without converting, specify a source buffer length of 0.

The caller can put the conversion data in any data space. To allow the conversion service to access the data, an ALET must be specified. An ALET of 0 indicates that the data is in the primary address space.

To indicate which code page was active at the end of conversions from and to mixed code pages, CUNBCPRM_Subcodepage is updated by the character conversion services. The same technique is used for designator sequences used for some ISO 2022 encoding.

Specifically, since an MBCS encoding is made up of SBCS and DBCS tables, a unique algorithm is used to deal with this in the character conversion service. When converting to an MBCS encoding, the character conversion service will first begin using the SBCS table to search for the character to be converted. If the code point is not in the valid range within the SBCS table (from X'00' to X'FF'), the conversion service will switch to the DBCS table to look for that code point and convert. It is that switch that will generate a X'0E' (Shift-Out) in the converted data stream, because a shift out of SBCS mode was performed. Next the character conversion service will continue using the DBCS table for subsequent conversions of characters. At this point, if there are no more characters to be converted, the character conversion service will stop the conversion and the converted data stream will end without a X'0F' (shift into SBCS mode). However, if the character conversion service encounters a code point that is in the valid SBCS code point range, the character conversion service will switch back to SBCS and thereby generating a X'0F' (Shift In) in the converted data stream, because a shift into SBCS mode was performed. It is the responsibility of the character conversion service exploiter to add the necessary SI/SO (Shift In/Shift Out) characters when a string is broken up across multiple calls to the character conversion service that involves MBCS characters.

This is where the CUNBCPRM_Subcodepage parameter is useful. CUNBCPRM_Subcodepage is made up of two halves - first half is CUNBCPRM_Source_SCP_State and second half is CUNBCPRM_Target_SCP_State. When converting from Unicode to EBCDIC(MBCS), the character conversion service will set CUNBCPRM_Target_SCP_State. When converting from EBCDIC(MBCS) to Unicode, the character conversion service will set CUNBCPRM_Source_SCP_State. See the "Description of parameters in area CUNBCPRM" on page 36 for the specific values and their definitions.

For the internal handling of MBCS conversions, refer to Appendix A, "MBCS conversions," on page 217.

Restrictions for the calling environment

Table 5. Restrictions while calling the character conversion services

Property	Restriction
Authorization	Problem state or supervisor state, and any PSW key
Dispatchable unit mode	Task or SRB
Cross memory mode	Any PASN, any HASN, any SASN
AMODE	31-bit and 64-bit
ASC mode	Called in primary mode but exploiting AR mode
Interrupt status	Enabled for I/O and external interrupts
Locks	May be held by the caller, but is not required to hold any
Control parameters	Must be in the primary address space
Recovery environment	Provided exclusively by the caller of the conversion services

Using the C interface

This is the call syntax in C for calling the stub routine **CUNLCNV** (character conversion). The mapping of the parameter area supplied by the header file `cunhc.h` is listed in “Mapping of parameters in C.” A sample program, `CUNSCSMC`, is provided in `SYS1.SAMPLIB`.

```
#include<cunhc.h>
#define SLEN 1000
#define WLEN 1000
#define TLEN 4096
.....
unsigned char Sourcebuffer [SLEN ];
unsigned char Targetbuffer [TLEN ];
unsigned char Workbuffer [WLEN ];
unsigned char DDA [CUNBCPRM_DDA_REQ ];

CUNBCPRM myparm ={CUNBCPRM_DEFAULT};
myparm.Src_Buf_Ptr=Sourcebuffer;
myparm.Targ_Buf_Ptr=Targetbuffer;
myparm.Targ_Buf_Len=TLEN;
myparm.Src_Buf_Len=SLEN;
myparm.Src_CCsid=850;
myparm.Targ_CCsid=1047;
memcpy(myparm.Technique,"LMER",4);
myparm.Wrk_Buf_Ptr=Workbuffer;
myparm.Wrk_Buf_Len=WLEN;
myparm.DDA_Buf_Ptr=DDA;
myparm.DDA_Buf_Len=CUNBCPRM_DDA_REQ;
CUNLCNV ( & myparm );
if((myparm.Return_Code !=CUN_RC_OK).....
```

Mapping of parameters in C

A C header file is supplied (`cunhc.h`) which contains the function prototypes for the conversion services. The following structures used in the interface to the character conversion service show the parameter list (`tagCUNBCPRM`) and conversion handle within the parameter list (`uccehdl`):

31-bit mapping

```

typedef struct tagCUNBCPRM {
    long      Version;           /* Structure version number */
    long      Length;           /* Length of structure */
    long      Res1;             /* Reserved */
    void *    Src_Buf_Ptr;       /* Pointer to Source */
    unsigned long Src_Buf_ALET;   /* ALET of source buffer */
    unsigned long Src_Buf_Len;   /* Length of source data */
    long      Res2;             /* Reserved */
    void *    Targ_Buf_Ptr;       /* Pointer to Target */
    unsigned long Targ_Buf_ALET; /* ALET of target buffer */
    unsigned long Targ_Buf_Len;  /* Length of target buffer */
    char      Conv_Handle[64];   /* conversion handle */
    unsigned long Src_CCSID;      /* CCSID of source data */
    unsigned long Targ_CCSID;     /* CCSID of target data */
    char      Technique[8];      /* */
    long      Res4;             /* Reserved */
    void *    Wrk_Buf_Ptr;       /* Pointer to work buffer */
    unsigned long Wrk_Buf_ALET;   /* ALET of work buffer */
    unsigned long Wrk_Buf_Len;   /* Length of work buffer */
    long      Res5;             /* Reserved */
    void *    DDA_Buf_Ptr;       /* Pointer to dynamic data area */
    unsigned long DDA_Buf_ALET;   /* ALET of DDA */
    unsigned long DDA_Buf_Len;   /* Length of DDA */
    struct {
        int      Sub_Action      : 1, /* Sub action: */
        /* 0 = Terminate with error */
        /* 1 = Substitute and cont. */
        int      Inv_Handle      : 1, /* Invalid handle at start: */
        /* 0 = Terminate with error */
        /* 1 = Get new handle and */
        int      No_Opt_Buf_Fill : 1, /* Target buffer filled */
        /* 0 = Target buffer filled */
        /* optimally increases runtime */
        /* 1 = Target Buffer not filled */
        /* optimally increases runtime */
        int      Mal_Action      : 1, /* Mal Action: (Default 0) */
        /* 0 = Substitute and cont. */
        /* 1 = Terminate with error */
        int      RL_Sub_Action   : 1, /* RL Sub action. If Tech=R/L: */
        /* 0 = Does nothing. */
        /* 1 = Override SUB_ACTION. */
        int      SrcSub_Chk      : 1, /* If Sub is checked: */
        /* 0=Does nothing. */
        /* 1=Override SUB_ACTION. */
        int      Bidi_Context    : 1, /* Bidi Context */
        /* 0 = Context LTR */
        /* 1 = Context RTL */
        int      Bidi_ImpAlg     : 1; /* Bidi Implicit Alg */
        /* 0 = Algor Basic */
        /* 1 = Algor Implicit */
    } Flag1;
    struct {
        int      Source_SCP_State :4, /*Source subcodepage state */
        int      Target_SCP_State :4; /*Target subcodepage state */
    } Subcodepage;
    struct {
        int      Substitution    : 1, /* Substitution: */
        /* 0 = No character substituted */
        /* 1 = character(s) substituted */
        int      Mal_Found       : 1, /* Malformed String found */
        /* 0 = No Malformed str found */
        /* 1 = Malformed str found */
        int      Page_Fix        : 1, /* Page fixing: */
        /* 0=System storage */
    }

```



```

/* 1=Page Fixing. */
ETF3E_Behavior_Status : 1, /* HW Enhancement for conver- */
/* sions from 1200 to 1208 and */
/* vice versa status: */
/* 0 = When ETF3E_Behavior is */
/* ON, means ETF3 HW enhancement*/
/* is used (default) */
/* 1 = When ETF3E_Behavior is */
/* ON, means ETF3 HW enhancement*/
/* is not used, because it is */
/* not available. */
/* */
/* Note. When conversion are not*/
/* requested from 1200 to */
/* 1208 and vice versa, the */
/* contents of this is not */
/* meaningful. */
: 4;

} Flag2;
unsigned char Designator; /* reserved for ISO 2022 */
long Return_Code;
long Reason_Code;
unsigned int Res4; /* Reserved */
struct {
int ETF3E_Behavior : 1, /* ETF3 HW Enhancement */
/* 0 = Do not exploit ETF3 */
/* (default) */
/* 1 = Exploit ETF3 */
: 15; /* Reserved */
} Flag3;
char Res5[2]; /* FLAG3 Byte 2 set by caller */
} CUNBCPRM;

```

64-bit mapping

```

typedef struct tagCUN4BCPR {
unsigned int Version; /* Structure version number */
unsigned int Length; /* Length of structure */
void * Src_Buf_Ptr; /* Pointer to Source */
unsigned int Src_Buf_ALET; /* ALET of source buffer */
unsigned int Res1; /* Reserved */
unsigned long Src_Buf_Len; /* Length of source data */
void * Targ_Buf_Ptr; /* Pointer to Target */
unsigned int Targ_Buf_ALET; /* ALET of target buffer */
unsigned int Res2; /* Reserved */
unsigned long Targ_Buf_Len; /* Length of target buffer */
char Conv_Handle[64]; /* conversion handle */
unsigned int Src_CCSID; /* CCSID of source data */
unsigned int Targ_CCSID; /* CCSID of target data */
char Technique[8]; /* */
void * Wrk_Buf_Ptr; /* Pointer to work buffer */
unsigned int Wrk_Buf_ALET; /* ALET of work buffer */
unsigned int Res3; /* Reserved */
unsigned long Wrk_Buf_Len; /* Length of work buffer */
void * DDA_Buf_Ptr; /* Pointer to dynamic data area */
unsigned int DDA_Buf_ALET; /* ALET of DDA */
unsigned int DDA_Buf_Len; /* Length of DDA */
struct {
int Sub_Action : 1, /* Sub action: */
/* 0 = Terminate with error */
/* 1 = Substitute and cont. */
Inv_Handle : 1, /* Invalid handle at start: */
/* 0 = Terminate with error */
/* 1 = Get new handle and */
No_Opt_Buf_Fill : 1, /* Target buffer filled */

```

```

/* 0 = Target buffer filled */
/* optimally increases runtime */
/* 1 = Target Buffer not filled*/
/* optimally increases runtime */
Mal_Action      : 1, /* Mal Action: (Default 0) */
/* 0 = Substitute and cont. */
/* 1 = Terminate with error */
RL_Sub_Action   : 1, /* RL Sub action. If Tech=R/L: */
/* 0 = Does nothing. */
/* 1 = Override SUB_ACTION. */
SrcSub_Chk      : 1, /* If Sub is checked: */
/* 0=Does nothing. */
/* 1=Override SUB_ACTION. */
Bidi_Context    : 1, /* Bidi Context */
/* 0 = Context LTR */
/* 1 = Context RTL */
Bidi_ImpAlg     : 1; /* Bidi Implicit Alg */
/* 0 = Algor Basic */
/* 1 = Algor Implicit */

} Flag1; /* FLAG Byte 1 set by caller */
struct { /* Subcodepage number(s) */
int Source_SCP_State :4, /*Source subcodepage state */
Target_SCP_State :4; /*Target subcodepage state */
} Subcodepage;
struct {
int Substitution : 1, /* Substitution: */
/* 0 = No character substituted*/
/* 1 = character(s) substituted*/
Mal_Found : 1, /* Malformed String found */
/* 0 = No Malformed str found */
/* 1 = Malformed str found */
Page_Fix : 1, /* Page fixing: */
/* 0=System storage */
/* 1=Page Fixing. */
ETF3E_Behavior_Status : 1, /* HW Enhancement for conver- */
/* sions from 1200 to 1208 and */
/* vice versa status: */
/* 0 = When ETF3E_Behavior is */
/* ON, means ETF3 HW enhancement*/
/* is used (default) */
/* 1 = When ETF3E_Behavior is */
/* ON, means ETF3 HW enhancement*/
/* is not used (because is */
/* not available) */
/* */
/* Note. When conversion are not*/
/* requested from 1200 to */
/* 1208 and vice versa, the */
/* contents of this is not */
/* meaningful. */
: 4;

} Flag2;
unsigned char Designator; /* reserved for ISO 2022 */
unsigned int Return_Code;
unsigned int Reason_Code;
long Res6; /* Reserved */
struct {
int ETF3E_Behavior : 1, /* ETF3 HW Enhancement */
/* 0 = Do not exploit ETF3 */
/* (default) */
/* 1 = Exploit ETF3 */
: 15; /* Reserved */
} Flag3; /* FLAG3 Byte 2 set by caller */
char Res7[2]; /* Reserved */
} CUN4BCPR;

```

Using the HLASM interface

This is the call syntax in HLASM for calling the stub routine **CUNLCNV** (character conversion for AMODE (31)) and **CUN4LCNV** (character conversion for AMODE (64)). A sample program, CUNSCSMA, is provided in SYS1.SAMPLIB.

For AMODE (31)

```
-----1-----2-----3-----4-----5-----6-----7--
      GETMAIN .....      Obtain storage for parameter area
*                               in primary address space
      LR   R4,R1           Save parameter area address
      USING CUNBCPRM,R4    Make parameter area addressable
      XC   CUNBCPRM(CUNBCPRM_LEN),CUNBCPRM   Init ParmArea to zero
      LA   R15,CUNBCPRM_VER   Get Version
      ST   R15,CUNBCPRM_VERSION Version Store to parameter area
      LA   R15,CUNBCPRM_LEN   Initialize Length
      ST   R15,CUNBCPRM_LENGTH Move to parameter area
      MVC  CUNBCPRM_TECHNIQUE,=CL8' ' Take default technique
      MVC  CUNBCPRM_SRC_CCSDID,=FL4'1047'   From CCSID
      MVC  CUNBCPRM_TARG_CCSDID,=FL4'13488' To CCSID

*
*      Supply source buffer pointer, length and ALET.
*      Supply target buffer pointer, length and ALET.
*      Supply work buffer pointer, length and ALET. (Not required
*      for a conversion from 1047 to 13488).
*      Supply DDA buffer pointer, length and ALET.
*      Note: A DDA is always required. The required DDA length is
*      defined by constant CUNBCPRM_DDA_REQ.
*
      CALL CUNLCNV,((R4)) Call stub routine with CUNBCPRM
*                               address as argument.
      CUNBCIDF DSECT=YES    Provide Mappings (CUNBCPRM, return and
*                               reason codes, constants for version
*                               and length).
```

For AMODE (64)

```
-----1-----2-----3-----4-----5-----6-----7--
      GETMAIN .....      Obtain storage for parameter area
*                               in primary address space
      LR   R4,R1           Save parameter area address
      USING CUN4BCPR,R4    Make parameter area addressable
      XC   CUN4BCPR,CUN4BCPR   Init PARAMETER AREA TO BINARY 0
      LA   R15,CUN4BCPR_VER   Get Version
      ST   R15,CUN4BCPR_VERSION Version Store to parameter area
      LA   R15,CUN4BCPR_LEN   Initialize Length
      ST   R15,CUN4BCPR_LENGTH Move to parameter area
      MVC  CUN4BCPR_TECHNIQUE,=CL8' ' Take default technique
      MVC  CUN4BCPR_SRC_CCSDID,=FL4'1047'   From CCSID
      MVC  CUN4BCPR_TARG_CCSDID,=FL4'13488' To CCSID

*
*      Supply source buffer pointer, length and ALET.
*      Supply target buffer pointer, length and ALET.
*      Supply work buffer pointer, length and ALET. (Not required
*      for a conversion from 1047 to 13488).
*      Supply DDA buffer pointer, length and ALET.
*
      CALL CUN4LCNV,((R4)) Call stub routine with CUN4BCPR
*                               address as argument.
      CUN4BCID DSECT=YES    Provide Mappings (CUN4BCPR, return and
*                               reason codes, constants for version
*                               and length).
```

Mapping of parameters for AMODE (31)

The mapping of the parameter areas are supplied by the interface definition file CUNBCIDF. This file is shipped in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that may be necessary.

Table 6. Mapping of parameters in HLASM for character conversion AMODE (31)

Offset Dec	Offset Hex	Type	Length in bytes	Boundary	Name	Description
0	(0)	STRUCTURE	169	DWORD	CUNBCPRM	Parameter
0	(0)	UNSIGNED	4		CUNBCPRM_Version	Parameter area VERSION
4	(4)	UNSIGNED	4		CUNBCPRM_Length	Parameter area Length
8	(8)	CHARACTER	4		*	Reserved for 64 bit
12	(C)	ADDRESS	4		CUNBCPRM_Src_Buf_Ptr	Source buffer pointer
16	(10)	UNSIGNED	4		CUNBCPRM_Src_Buf_ALET	Source buffer ALET
20	(14)	UNSIGNED	4		CUNBCPRM_Src_Buf_Len	Source buffer length
24	(18)	CHARACTER	4		*	Reserved for 64 bit
28	(1C)	ADDRESS	4		CUNBCPRM_Targ_Buf_Ptr	Target buffer pointer
32	(20)	UNSIGNED	4		CUNBCPRM_Targ_Buf_ALET	Target buffer ALET
36	(24)	UNSIGNED	4		CUNBCPRM_Targ_Buf_Len	Target buffer length
40	(28)	CHARACTER	64	DWORD	CUNBCPRM_Conv_Handle	Conversion handle
104	(68)	CHARACTER	16	WORD	CUNBCPRM_Conv_Key	Conversion Key
104	(68)	UNSIGNED	4		CUNBCPRM_Src_CCSID	Source CCSID (codepage)
108	(6C)	UNSIGNED	4		CUNBCPRM_Targ_CCSID	Target CCSID (codepage)
112	(70)	CHARACTER	8		CUNBCPRM_Technique	The CONVERSION TECHNIQUE is specified as input to the image generator
120	(78)	CHARACTER	4		*	Reserved for 64 bit
124	(7C)	ADDRESS	4		CUNBCPRM_Wrk_Buf_Ptr	Work buffer pointer
128	(80)	UNSIGNED	4		CUNBCPRM_Wrk_Buf_ALET	Work buffer ALET
132	(84)	UNSIGNED	4		CUNBCPRM_Wrk_Buf_Len	Work buffer length
136	(88)	CHARACTER	4		*	Reserved for 64 bit
140	(8C)	ADDRESS	4		CUNBCPRM_DDA_Buf_Ptr	Dynamic data area pointer
144	(90)	UNSIGNED	4		CUNBCPRM_DDA_Buf_ALET	Dynamic data area ALET
148	(94)	UNSIGNED	4		CUNBCPRM_DDA_Buf_Len	Dynamic data area length as defined by constant CUNBCPRM_DDA_Req

Table 6. Mapping of parameters in HLASM for character conversion AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in bytes	Boundary	Name	Description
152	(98)	BITSTRING	1		CUNBCPRM_Flag1	FLAG Byte 1 set by caller
		1...			CUNBCPRM_Sub_Action	Sub action: 0=TERMINATE WITH ERROR 1=Substitute AND CONT
		.1..			CUNBCPRM_Inv_Handle	Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONT
		 1..			CUNBCPRM_RL_Sub_Action	R or L technique action
		1..			CUNBCPRM_SrcSub_Chk	Substitution Chars Check in source: 0=Do nothing 1=Override SUB_ACTION
		1.			CUNBCPRM_Bidi_Context	Bidi Context: 0=Context LTR 1=Context RTL
		1			CUNBCPRM_Bidi_ImpAlg	Bidi Implicit Alg: 0=Algor Basic 1=Algor Implicit
153	(99)	UNSIGNED	1		CUNBCPRM_Subcodepage	Number of subcodepage(s)
		BITSTRING 1111			CUNBCPRM_Source_SCP_State	Source subcodepage status
		BITSTRING 1111			CUNBCPRM_Target_SCP_State	Target subcodepage status

Table 6. Mapping of parameters in HLASM for character conversion AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in bytes	Boundary	Name	Description
154	(9A)	BITSTRING	1		CUNBCPRM_Flag2	FLAG Byte 2 set by service
		1...			CUNBCPRM_Substitution	Substitution: 0=NO CHARACTER SUBSTITUTED 1=CHARACTER(S) SUBSTITUTED.
		.1..			CUNBCPRM_Mal_Found	Malformed String found: 0=NO MALFORMED STRING FOUND 1=MALFORMED STRING FOUND.
		..1.			CUNBCPRM_Page_Fix	Page fixing: 0=System storage 1=Page Fixing
		...1			CUNBCPRM ETF3E_Behavior_ Status	ETF3 hardware enhancement for conversions from 1200 to 1208 and vice versa. The meanings of the values are: 0=ETF3 hardware enhancement is enabled. 1=ETF3 hardware enhancement is not installed.
155	(9B)	UNSIGNED	1		CUNBCPRM_Designator	Reserved for ISO2022
156	(9C)	CHARACTER	8	WORD	CUNBCPRM_RC_RS	Return/reason code
156	(9C)	UNSIGNED	4		CUNBCPRM_Return_Code	Return code
160	(A0)	UNSIGNED	4		CUNBCPRM_Reason_Code	Reason code
164	(A4)	BITSTRING	1		CUNBCPRM_Flag3	FLAG Byte 3 set by caller
		1...			CUNBCPRM ETF3E_Behavior	ETF3 hardware enhancement implementation for conversions from 1200 to 1208 and vice versa: 0=Do not exploit ETF3 hardware enhancement. 1=Exploit ETF3 hardware enhancement.
165	(A5)		0		CUNBCPRM_End	End of CUNBCPRM

Description of parameters in area CUNBCPRM

This description applies to C and HLASM.

CUNBCPRM_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUNLCNV using the constant CUNBCPRM_Ver which is supplied by the interface definition file CUNBCIDF.

The parameter CUNBCPRM_Version2 is defined to exploit the extended-translation facility 3 (ETF3) function.

CUNBCPRM_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUNLCNV using the constant CUNBCPRM_Len which is supplied by the interface definition file CUNBCIDF.

CUNBCPRM_Src_Buf_Ptr - set by caller

Specifies the beginning address of a string of text characters encoded in the CCSID named in the CUNBCPRM_Src_CCSID parameter, and with a length specified in the CUNBCPRM_Src_Buf_Len parameter. At the completion of the conversion, CUNBCPRM_Src_Buf_Ptr will be updated to point just past the last character that was successfully converted, and CUNBCPRM_Src_Buf_Len will be updated to reflect the number of bytes left unconverted. If all bytes are converted, CUNBCPRM_Src_Buf_Len will be zero.

CUNBCPRM_Src_Buf_ALET - set by caller

Specifies the ALET to be used if the source buffer addressed by CUNBCPRM_Src_Buf_ptr resides in a different address or data space.

CUNBCPRM_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer addressed by CUNBCPRM_Src_Buf_Ptr, to be converted. The source buffer length may be zero. In this case, nothing is converted but the CUNBCPRM_Conv_Handle is returned. This may be used to request a handle without converting. The maximum allowed value is X'FFFFFFFF'.

CUNBCPRM_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage where the converted text string will be stored. At the completion of the conversion, CUNBCPRM_Targ_Buf_Ptr will point just past the last character stored, and CUNBCPRM_Targ_Buf_Len will be updated to indicate the number of bytes not yet consumed in the buffer.

CUNBCPRM_Targ_Buf_ALET - set by caller

Specifies the ALET to be used, if the target buffer addressed by CUNBCPRM_Targ_Buf_Ptr resides in a different address or data space.

CUNBCPRM_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUNBCPRM_Targ_Buf_Ptr. At any point during a conversion, this length must be able to hold at least one character of the maximum width for the specified TO-CCSID (target code page) whenever CUNBCPRM_Src_Buf_Len is greater than 0. The maximum allowed value is X'FFFFFFFF'.

CUNBCPRM_Conv_Handle - set by conversion service

Specifies the handle to a UCCE. If a handle is present it will be used, otherwise the CUNBCPRM_Src_CCSID, CUNBCPRM_Targ_CCSID, and CUNBCPRM_Technique (if provided) parameters will be used and a handle to UCCE is returned in CUNBCPRM_Conv_Handle. Subsequent calls to stub routine CUNLCNV, requesting the same conversion, will be faster because the handle is used and CUNBCPRM_Conv_Handle does not need to be recomputed.

Note: For the first call to stub routine CUNLCNV, CUNBCPRM_Conv_Handle must be set to binary zero X'00'.

CUNBCPRM_Conv_Key

Specifies a structure that can be used to access CUNBCPRM_Src_CCSID, CUNBCPRM_Targ_CCSID, and CUNBCPRM_Technique as one unit.

CUNBCPRM_Src_CCSID - set by caller, updated by service*

Specifies the CCSID encoding of the text in the source buffer. The contents of CUNBCPRM_Src_CCSID must be a valid CCSID. It must correspond to the CUNBCPRM_Targ_CCSID parameter so that there is a valid UCCD built during IPL and it may be changed by a SET UNI command. This parameter is mandatory for the first call to stub routine CUNLCNV. It is not used if a non-zero CUNBCPRM_Conv_Handle is given.

Note: When CCSID 1200 is specified this parameter will be updated by the service accordingly with the Unicode version supported for this conversion. See “Control statement CONVERSION” on page 11 for some special considerations about CCSID 1200, for the list of UCS-2 CCSIDs versions supported.

CUNBCPRM_Targ_CCSID - set by caller, updated by service*

Specifies the CCSID encoding of the text in the target buffer. The contents of CUNBCPRM_Targ_CCSID must be a valid CCSID. It must correspond with the CUNBCPRM_Src_CCSID parameter in a way that there is a valid UCCE built during IPL and this may be changed by a SET UNI command. This parameter is mandatory for the first call to CUNLCNV. It is not used if a non-zero CUNBCPRM_Conv_Handle is given.

Note: When CCSID 1200 is specified this parameter will be updated by the service accordingly with the Unicode version supported for this conversion. See “Control statement CONVERSION” on page 11 for some special considerations about CCSID 1200, for the list of UCS-2 CCSIDs versions supported.

CUNBCPRM_Technique - set by caller

Specifies the technique-search-order for the given CCSID pair. See “Understanding how Unicode Services loads conversion tables” on page 20. In addition to the techniques search orders (R,E,C,L,M and 0-9) that are supported currently, you can also use technique “B” to invoke BIDI service through Character Conversion Service API. When technique “B” is requested, target buffer will contain the to-CCSID conversion plus BIDI properties. Consider the following characteristics when you use technique “B”:

- Technique “B” can be combined in any order with the current supported techniques search orders (R,E,C,L,M, and 0-9).
- When technique “B” is requested, CUNBCPRM_DDA_Req2 must be used as DDA value for CUNBCPRM_DDA_Buf_Len.
- Technique “B” is not supported by the Image generator CUNMIUTL.
- Technique “B” is not part of the default technique search order RECLM.
- Technique “B” is not supported through the SETUNI command.

CUNBCPRM_Wrk_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion services can use to store intermediate results.

CUNBCPRM_Wrk_Buf_ALET - set by caller

Specifies the ALET to be used if the work buffer addressed by CUNBCPRM_Wrk_Buf_Ptr resides in a different address or data space.

CUNBCPRM_Wrk_Buf_Len - set by caller

Specifies the length in bytes of the work buffer addressed by CUNBCPRM_Wrk_Buf_Ptr. The parameter CUNBCPRM_Wrk_Buf_Len must be equal or greater than 2 if CUNBCPRM_Src_Buf_Len is greater

than 0. A work buffer is only required for indirect conversions. See “Calling the character conversion services” on page 25. The maximum allowed value is X'7FFFFFFF'.

CUNBCPRM_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion services are using internally as dynamic data area.

Note: CUNBCPRM_DDA_Buf_Ptr must be double-word boundary.

CUNBCPRM_DDA_Buf_ALET - set by caller

Specifies the ALET to be used if the dynamic data area addressed by CUNBCPRM_DDA_Buf_Ptr resides in a different address or data space.

CUNBCPRM_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUNBCPRM_DDA_Buf_Ptr. The required length is defined by constant CUNBCPRM_DDA_Req.

Note: When technique “B” is requested, use CUNBCPRM_DDA_Req2 to specify the CUNBCPRM_DDA_Buf_Len. You can find CUNBCPRM_DDA_Req2 in the Character Conversion Interface Definition File CUNBCIDF.

CUNBCPRM_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUNBCPRM_Sub_Action
x1xx xxxx	CUNBCPRM_Inv_Handle
xx1x xxxx	CUNBCPRM_No_Opt_Buf_Fill
xxx1 xxxx	CUNBCPRM_Ma1_Action
xxxx 1xxx	CUNBCPRM_RL_Sub_Action
xxxx x1xx	CUNBCPRM_SrcSub_Chk
xxxx xx1x	CUNBCPRM_Bidi_Context
xxxx xxx1	CUNBCPRM_Bidi_ImpAlg

CUNBCPRM_Sub_Action

Specifies the action to take when either a source character that is not convertible to the TO-CCSID or the substitution character in the FROM-CCSID is encountered.

- **0:** Indicates that the conversion is to be terminated with an error.
- **1:** Indicates that the substitution character is to be put in the target buffer and the conversion is to be continued.

CUNBCPRM_Inv_Handle

Specifies what has to be done when the UCCE handle is invalid.

- **0:** Indicates that the conversion is to be terminated with return code CUN_RC_WARN and reason code CUN_RS_INV_HANDLE_SET or CUN_RS_INV_HANDLE_NOSET.
- **1:** Indicates that the conversion is to be done with a new handle created by the conversion services and put into CUNBCPRM_Conv_Handle. This is done only if no SET UNI or SETUNI command is running. If the SET UNI command is still

running, the conversion will be terminated with return code CUN_RC_WARN and reason code CUN_RS_INV_HANDLE_SET.

CUNBCPRM_No_Opt_Buf_Fill

Specifies whether the target buffer is to be filled to a maximum for indirect conversion. This bit enables the caller to choose between fast execution without an optimally filled target buffer, and slower execution, but with a target buffer optimally filled.

- **0:** Indicates that the target buffer is to be filled to a maximum, taking additional steps into account. The benefit is that the target buffer is always filled with as many characters as possible, although processing time may be slow.
- **1:** Indicates that the target buffer is not filled to a maximum, which may decrease processing time. However, the number of characters that fit into the target buffer is only estimated once. Therefore, characters may be left in the source buffer, although the corresponding target characters would fit into the target buffer.

CUNBCPRM_Mal_Action

Specifies the action to be taken when a source character is malformed on the source CCSID.

Note: This action only takes place when CUNBCPRM_Sub_Action is 1.

- **0:** Indicates that the substitution character is to be put in the target buffer, and the conversion is to be continued.
- **1:** Indicates that the conversion is to be terminated with return code CUN_RC_WARN and reason code CUN_RS_MAL_CHAR_ACT_TERM.

CUNBCPRM_RL_Sub_Action

Specifies what has to be done when "R" or "L" techniques are specified in the conversion call when a substitution character is converted.

- **0:** Indicates that CUNBCPRM_Sub_Act will work normally.
- **1:** Indicates that CUNBCPRM_Sub_Act will be overridden to 0 and no substitution bit (CUNBCPRM_Substitution) will be flagged.

CUNBCPRM_SrcSub_Chk

Specifies whether the service will consider source substitution chars as substitution or not.

- **0:** Indicates that the substitution character was placed in the target buffer when one or more malformed, invalid or substitution character were found in the source. In addition, the CUNBCPRM_Substitution flag, part of CUNBCPRM_Flag2, is turned on.
- **1:** Indicates that when a substitution character belonging to the FROM-CCSID is found in the source, a substitution character is placed in the target buffer, but the CUNBCPRM_Substitution flag is not turned on.

Note: This action only takes place when CUNBCPRM_Sub_Action is 1. In addition, it is highly recommended that exploiters of

this bit, notify their customers to rebuild their images, to avoid a degradation in performance.

CUNBCPRM_Bidi_Context

Specifies the context of the text to be transformed with the bidi service if technique B was specified.

- **0**: Indicates the context is Left to Right (LTR).
- **1**: Indicates the context is Right to Left (RTL).

CUNBCPRM_Bidi_ImpAlg

Specifies the algorithm to be used if technique B was specified.

- **0**: Indicates the basic algorithm will be used.
- **1**: Indicates the implicit algorithm will be used.

For more information, see Chapter 8, "Bidi transformation," on page 155.

CUNBCPRM_Subcodepage - set by caller initially, then set by conversion service

Used for conversions with CCSIDs that have a "state-dependent" encoding scheme (such as EBCDIC MBCS). For each new source string, on the first call to the character conversion service, CUNBCPRM_Subcodepage should be set to zero. Thus the converter will start with default subcodepage(s). When the conversion service returns, CUNBCPRM_Subcodepage is updated to reflect the subcode page number used when converting the last source character. For subsequent calls to the character conversion service, (partial string processing of long source strings), CUNBCPRM_Subcodepage must be used unchanged as returned from the previous call. Thus the next piece of source will start with the correct subcode page.

CUNBCPRM_Subcodepage is made up of two halfbytes. The first halfbyte can be referenced by the name CUNBCPRM_Source_SCP_State. The second halfbyte can be referenced by the name CUNBCPRM_Target_SCP_State.

CUNBCPRM_Source_SCP_State - set by caller initially, then set by conversion service

Reflects the From_CCSID's subcode page used for the last converted character. Specifically, CUNBCPRM_Source_SCP_State is set to:

0	To denote that a 'non-state' dependent' encoding scheme was used.
1	To denote that the last character converted came from an SBCS EBCDIC table.
2	To denote that the last character converted came from a DBCS EBCDIC table.
3	To denote that the last character converted came from a TBCS EBCDIC table.
4	To denote that the last character converted came from a QBCS EBCDIC table.
5	To denote that the last character converted came from an SBCS ASCII table.

- 6 To denote that the last character converted came from a DBCS ASCII table.
- 7 To denote that the last character converted came from a TBCS ASCII table.
- 8 To denote that the last character converted came from a QBCS ASCII table.

An easy way to get the value of this halfbyte is to 'AND' CUNBCPRM_Subcodepage with 'F0'.

CUNBCPRM_Target_SCP_State - set by caller initially, then set by conversion service

Reflects the TO-CCSID's subcodepage used for the last converted character. Specifically, CUNBCPRM_Target_SCP_State is set to:

- 0 To denote that a 'non-state dependent' encoding scheme was used.
- 1 To denote that the last character converted came from an SBCS EBCDIC table.
- 2 To denote that the last character converted came from a DBCS EBCDIC table.
- 3 To denote that the last character converted came from a TBCS EBCDIC table.
- 4 To denote that the last character converted came from a QBCS EBCDIC table.
- 5 To denote that the last character converted came from an SBCS ASCII table.
- 6 To denote that the last character converted came from a DBCS ASCII table.
- 7 To denote that the last character converted came from a TBCS ASCII table.
- 8 To denote that the last character converted came from a QBCS ASCII table.

An easy way to get the value of this halfbyte is to 'AND' CUNBCPRM_Subcodepage with '0F'.

For example, when converting from MBCS to Unicode (UCS-2 or UTF-8) or any non-MBCS CCSID, CUNBCPRM_Source_SCP_State will be set. When converting from Unicode (UCS-2 or UTF-8) or any non-MBCS CCSID to MBCS, CUNBCPRM_Target_SCP_State will be set. When converting from any MBCS CCSID to another MBCS CCSID, both CUNBCPRM_Source_SCP_State and CUNBCPRM_Target_SCP_State will be set.

CUNBCPRM_Designator - set by conversion service

The parameter CUNBCPRM_Designator is used for conversions from and to ISO2022 encodings that use designator sequence. It specifies the active designator sequence in which the conversion is to begin. When the service returns, CUNBCPRM_Designator is updated as appropriate to reflect designator sequence active at the completion of the conversion.

For conversions to ISO2022-KR, which use only one designator, the sequence value means:

- **0:** The designator sequence was not yet inserted
- **1:** The designator sequence was already inserted

CUNBCPRM_Flag2 - set by service and caller

Bit position	Name
1xxx xxxx	CUNBCPRM_Substitution
x1xx xxxx	CUNBCPRM_Mal_Found
xx1x xxxx	CUNBCPRM_Page_Fix
xxx1 xxxx	CUNBCPRM ETF3E_Behavior_Status

CUNBCPRM_Substitution

Indicates to the caller whether the conversion service has converted a character into the conversion table's substitution character.

Note: This bit has to be reset by the caller.

- **0:** Indicates that the conversion service did not substitute.
- **1:** Indicates that the conversion service converted at least 1 character into the conversion table's substitution character (or the service was already called with bit set to 1).

CUNBCPRM_Mal_Found

Indicates to the caller whether the conversion service has encountered a malformed character in the source buffer.

Note: This bit has to be reset by the caller.

- **0:** Indicates that the conversion service did not find a malformed character in the source buffer.
- **1:** Indicates that the conversion service found at least one malformed character in the source buffer (or the service was already called with bit set to 1).

CUNBCPRM_Page_Fix

Indicates in case the conversion is not loaded whether it will be loaded on a Page fixed basis or not.

- **0:** Indicates Conversion will not be loaded on Page Fix.
- **1:** Indicates Conversion will be loaded on Page Fix.

Note:

- This bit has to be reset by the caller.
- CUNBCPRM_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUNBCPRM ETF3E_Behavior_Status

The ETF3 and ETF3 enhancement are hardware features that can be used by the Unicode services to increase performance of certain translations between specific Unicode CCSIDs. For more information about these facilities, See *z/Architecture Principles of Operation*.

The bit CUNBCPRM ETF3E_Behavior_Status indicates the presence of the ETF3 enhancement facility. This bit is set to the appropriate value by the Unicode services. When

CUNBCPRM ETF3E_Behavior is ON, it indicates that whether the hardware enhancement is in use for conversions from 1200 to 1208 and vice versa.

Note: When the conversion is not requested from 1200 to 1208 and vice versa, the contents of this flag is not meaningful.

- **0:** Indicates that ETF3 hardware enhancement is used. 0 is the default.
- **1:** Indicates that ETF3 hardware enhancement is not installed.

CUNBCPRM_RC_RS

Specifies a structure that can be used to access CUNBCPRM_Return_Code and CUNBCPRM_Reason_Code as one unit.

CUNBCPRM_Return_Code - set by service

Specifies the return code.

CUNBCPRM_Reason_Code - set by service

Specifies the reason code.

CUNBCPRM_Flag3 - set by caller

Bit position	Name
1xxx xxxx	CUNBCPRM ETF3E_Behavior

CUNBCPRM ETF3E_Behavior

Specify whether to use the ETF3 hardware enhancement for conversions from 1200 to 1208 and vice versa.

Note: To make this flag meaningful, the parameter area version field CUNBCPRM_Version must be defined as CUNBCPRM_Version2; otherwise, this flag is ignored.

- **0:** Do not exploit ETF3 hardware enhancement. 0 is the default.
- **1:** Use ETF3 hardware enhancement.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas are supplied by the interface definition file CUN4BCID. This file is shipped in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that may be necessary.

Table 7. Mapping of parameters in HLASM for character conversion AMODE (64)

Offset Dec	Offset Hex	Type	Length in bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	193	DWORD	CUN4BCPR	Parameter
0	(0)	UNSIGNED	4		CUN4BCPR_Version	Parameter area VERSION
4	(4)	UNSIGNED	4		CUN4BCPR_Length	Parameter area Length
8	(8)	CHARACTER	8		CUN4BCPR_Src_Buf_Ptr	Source buffer pointer
16	(10)	ADDRESS	4		CUN4BCPR_Src_Buf_ALET	Source buffer ALET
20	(14)	UNSIGNED	4		*	Reserved
24	(18)	UNSIGNED	8		CUN4BCPR_Src_Buf_Len	Source buffer length
32	(20)	CHARACTER	8		CUN4BCPR_Targ_Buf_Ptr	Target buffer pointer

Table 7. Mapping of parameters in HLASM for character conversion AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in bytes	Boundary	Name	Short Description - See full description following table for details
40	(28)	ADDRESS	4		CUN4BCPR_Targ_Buf_ALET	Target buffer ALET
44	(2C)	UNSIGNED	4		*	Reserved
48	(30)	UNSIGNED	8		CUN4BCPR_Targ_Buf_Len	Target buffer length
56	(38)	CHARACTER	64	DWORD	CUN4BCPR_Conv_Handle	Conversion handle
120	(78)	CHARACTER	16	WORD	CUN4BCPR_Conv_Key	Conversion Key
120	(78)	UNSIGNED	4		CUN4BCPR_Src_CCSID	Source CCSID (codepage)
124	(7C)	UNSIGNED	4		CUN4BCPR_Targ_CCSID	Target CCSID (codepage)
128	(80)	CHARACTER	8		CUN4BCPR_Technique	The CONVERSION TECHNIQUE is specified as input to the image generator
136	(88)	CHARACTER	8		CUN4BCPR_Wrk_Buf_Ptr	Work buffer pointer
144	(90)	ADDRESS	4		CUN4BCPR_Wrk_Buf_ALET	Work buffer ALET
148	(94)	UNSIGNED	4		*	Reserved for 64 bit
152	(98)	UNSIGNED	8		CUN4BCPR_Wrk_Buf_Len	Work buffer length
160	(AC)	CHARACTER	4		*	Reserved for 64 bit
164	(A4)	ADDRESS	4		CUN4BCPR_DDA_Buf_Ptr	Dynamic data area pointer
168	(A8)	UNSIGNED	4		CUN4BCPR_DDA_Buf_ALET	Dynamic data area ALET
172	(AC)	UNSIGNED	4		CUN4BCPR_DDA_Buf_Len	Dynamic data area length as defined by constant CUN4BCPR_DDA_Req

Table 7. Mapping of parameters in HLASM for character conversion AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in bytes	Boundary	Name	Short Description - See full description following table for details
176	(B0)	BITSTRING	1		CUN4BCPR_Flag1	FLAG Byte 1 set by caller
		1...			CUN4BCPR_Sub_Action	Sub action: 0=TERMINATE WITH ERROR. 1=Substitute AND CONT.
		.1..			CUN4BCPR_Inv_Handle	Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONT.
		..1.			CUN4BCPR_No_Opt_Buf_Fill	Target buffer filled: 0=TARGET BUFFER FILLED OPTIMALLY. 1=TARGET BUFFER NOT FILLED OPTIMALLY.
		...1			CUN4BCPR_Mal_Action	Mal Action: (Default 0): 0=SUBSTITUTE AND CONT. 1=TERMINATE WITH ERROR
		 1...			CUN4BCPR_RL_Sub_Action	R or L technique action
		1..			CUN4BCPR_SrcSub_Chk	Substitution Chars Check in source: 0=Does nothing. 1=Override SUB_ACTION.
		1.			CUN4BCPR_Bidi_Context	Bidi Context: 0=Context LTR 1=Context RTL
		1			CUN4BCPR_Bidi_ImpAlg	Bidi Implicit Alg: 0=Algor Basic 1=Algor Implicit
177	(B1)	UNSIGNED	1		CUN4BCPR_Subcodepage	Number of subcodepage(s)
		BITSTRING 1111			CUN4BCPR_Source_SCP_State	Source subcodepage status
		BITSTRING 1111			CUN4BCPR_Target_SCP_State	Target subcodepage status

Table 7. Mapping of parameters in HLASM for character conversion AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in bytes	Boundary	Name	Short Description - See full description following table for details
178	(B2)	BITSTRING	1		CUN4BCPR_Flag2	FLAG Byte 2 set by service
		1...			CUN4BCPR_Substitution	Substitution: 0=NO CHARACTER SUBSTITUTED. 1=CHARACTER(S) SUBSTITUTED.
		.1..			CUN4BCPR_Mal_Found	Malformed string found: 0=NO MALFORMED STRING FOUND 1=MALFORMED STRING FOUND.
		..1.			CUN4BCPR_Page_Fix	Page fixing: 0=System storage 1=Page Fixing
		...1 ...			CUN4BCPR ETF3E_Behavior_ Status	ETF3 hardware enhancement for conversions from 1200 to 1208 and vice versa. When CUN4BCPR ETF3E_Behavior is on: 0=ETF3 hardware enhancement is enabled. 1=ETF3 hardware enhancement is not installed.
179	(B3)	UNSIGNED	1		CUN4BCPR_Designator	Reserved for ISO2022
180	(B4)	CHARACTER	8	WORD	CUN4BCPR_RC_RS	Return/reason code
		UNSIGNED	4		CUN4BCPR_Return_Code	Return code
		UNSIGNED	4		CUN4BCPR_Reason_Code	Reason code
188	(BC)	BITSTRING	1		CUN4BCPR_Flag3	FLAG3 Byte 3 set by caller
					CUN4BCPR ETF3E_Behavior	ETF3 hardware enhancement implementation for conversions from 1200 to 1208 and vice versa: 0=Do not exploit ETF3 hardware enhancement. 1=Exploit ETF3 hardware enhancement.
189	(BD)		0		CUN4BCPR_End	End of CUN4BCPR

Description of parameters in area CUN4BCPR

This description applies to HLASM.

CUN4BCPR_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUN4LCNV using the constant CUN4BCPR_Ver which is supplied by the interface definition file CUN4BCID.

The parameter CUN4BCPR_Version2 is defined to exploit the ETF3 hardware enhancement function.

CUN4BCPR_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUN4LCNV using the constant CUN4BCPR_Len which is supplied by the interface definition file CUN4BCID.

CUN4BCPR_Src_Buf_Ptr - set by caller

Specifies the first eight bytes of address of a string of text characters encoded in the CCSID named in the CUN4BCPR_Src_CCSID parameter, and with a length specified in the CUN4BCPR_Src_Buf_Len parameter. At the completion of the conversion, CUN4BCPR_Src_Buf_Ptr will be updated to point just past the last character that was successfully converted, and CUN4BCPR_Src_Buf_Len will be updated to reflect the number of bytes left unconverted. If all bytes are converted, CUN4BCPR_Src_Buf_Len will be zero.

CUN4BCPR_Src_Buf_ALET - set by caller

Specifies the ALET to be used if the source buffer addressed by CUN4BCPR_Src_Buf_Ptr resides in a different address or data space.

CUN4BCPR_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUN4BCPR_Src_Buf_Ptr, to be converted. The source buffer length can be zero. In this case, nothing is converted but the CUN4BCPR_Conv_Handle is returned. This can be used to request a handle without converting. The maximum allowed value is X'7FFFFFFFFFFFFFFF'.

CUN4BCPR_Targ_Buf_Ptr - set by caller

Specifies the first eight bytes of address of an area of storage where the converted text string will be stored. At the completion of the conversion, CUN4BCPR_Targ_Buf_Ptr will point just past the last character stored, and CUN4BCPR_Targ_Buf_Len will be updated to indicate the number of bytes not yet consumed in the buffer.

CUN4BCPR_Targ_Buf_ALET - set by caller

Specifies the ALET to be used, if the target buffer addressed by CUN4BCPR_Targ_Buf_Ptr resides in a different address or data space.

CUN4BCPR_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUN4BCPR_Targ_Buf_Ptr. At any point during a conversion, this length must be able to hold at least one character of the maximum width for the specified TO-CCSID (target code page) whenever CUN4BCPR_Src_Buf_Len is greater than 0. The maximum allowed value is X'7FFFFFFFFFFFFFFF'.

CUN4BCPR_Conv_Handle - set by conversion service

Specifies the handle to a UCCE. If a handle is present it will be used, otherwise the CUN4BCPR_Src_CCSID, CUN4BCPR_Targ_CCSID, and CUN4BCPR_Technique (if provided) parameters will be used and a handle to UCCE is returned in CUN4BCPR_Conv_Handle. Subsequent calls to stub routine CUN4LCNV, requesting the same conversion, will be faster because the handle is used and CUN4BCPR_Conv_Handle does not need to be recomputed.

Note: For the first call to stub routine CUN4LCNV, CUN4BCPR_Conv_Handle must be set to binary zero X'00'.

CUN4BCPR_Conv_Key

Specifies a structure that can be used to access CUN4BCPR_Src_CCSID, CUN4BCPR_Targ_CCSID, and CUN4BCPR_Technique as one unit.

CUN4BCPR_Src_CCSID - set by caller, updated by service*

Specifies the CCSID encoding of the text in the source buffer. The contents of CUN4BCPR_Src_CCSID must be a valid CCSID. It must correspond to the CUN4BCPR_Targ_CCSID parameter so that there is a valid UCCE built during IPL and it may be changed by a SET UNI command. This parameter is mandatory for the first call to stub routine CUN4LCNV. It is not used if a non-zero CUN4BCPR_Conv_Handle is given.

Note: When CCSID 1200 is specified this parameter will be updated by the service accordingly with the Unicode version supported for this conversion. See “Control statement CONVERSION” on page 11 for some special considerations about CCSID 1200, for the list of UCS-2 CCSIDs versions supported.

CUN4BCPR_Targ_CCSID - set by caller, updated by service*

Specifies the CCSID encoding of the text in the target buffer. The contents of CUN4BCPR_Targ_CCSID must be a valid CCSID. It must correspond with the CUN4BCPR_Src_CCSID parameter so that there is a valid UCCE built during IPL and this may be changed by a SET UNI command. This parameter is mandatory for the first call to CUN4LCNV. It is not used if a non-zero CUN4BCPR_Conv_Handle is given.

Note: When CCSID 1200 is specified this parameter will be updated by the service accordingly with the Unicode version supported for this conversion. See “Control statement CONVERSION” on page 11 for some special considerations about CCSID 1200, for the list of UCS-2 CCSIDs versions supported.

CUN4BCPR_Technique - set by caller

Specifies the technique-search-order for the given CCSID pair. See “Character conversion” on page 19. In addition to the techniques search orders (R,E,C,L,M, and 0-9) that are supported currently, you can also use technique “B” to invoke BIDI service through Character Conversion Service API. When technique “B” is requested, target buffer will contain the to-CCSID conversion plus BIDI properties. Consider the following characteristics when you use technique “B”:

- Technique “B” can be combined in any order with the current supported techniques search orders (R,E,C,L,M, and 0-9).
- When technique “B” is requested, CUN4BCPR_DDA_Req2 must be used as DDA value for CUN4BCPR_DDA_Buf_Len.
- Technique “B” is not supported by the Image generator CUNMIUTL.
- Technique “B” is not part of default technique search order RECLM.
- Technique “B” is not supported through the SETUNI command.

CUN4BCPR_Wrk_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion services can use to store intermediate results.

CUN4BCPR_Wrk_Buf_ALET - set by caller

Specifies the ALET to be used if the work buffer addressed by CUN4BCPR_Wrk_Buf_Ptr resides in a different address or data space.

CUN4BCPR_Wrk_Buf_Len - set by caller

Specifies the length in bytes of the work buffer addressed by CUN4BCPR_Wrk_Buf_Ptr. The parameter CUN4BCPR_Wrk_Buf_Len must be equal or greater than 2, if CUN4BCPR_Src_Buf_Len is greater than 0. A

work buffer is only required for indirect conversions. See “Calling the character conversion services” on page 25. The maximum allowed value is X'7FFFFFFFFFFFFFFF'.

CUN4BCPR_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion services are using internally as dynamic data area.

Note: CUN4BCPR_DDA_Buf_Ptr must be double-word boundary.

CUN4BCPR_DDA_Buf_ALET - set by caller

Specifies the ALET to be used, if the dynamic data area addressed by CUN4BCPR_DDA_Buf_Ptr resides in a different address or data space.

CUN4BCPR_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUN4BCPR_DDA_Buf_Ptr. The required length is defined by constant CUN4BCPR_DDA_Req.

Note: When technique “B” is requested, use CUN4BCPR_DDA_Req2 to specify the CUN4BCPR_DDA_Buf_Len. CUN4BCPR_DDA_Req2 is provided in the Character Conversion Interface Definition File CUN4BCID.

CUN4BCPR_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUN4BCPR_Sub_Action
x1xx xxxx	CUN4BCPR_Inv_Handle
xx1x xxxx	CUN4BCPR_No_Opt_Buf_Fill
xxx1 xxxx	CUN4BCPR_Mal_Action
xxxx 1xxx	CUN4BCPR_RL_Sub_Action
xxxx x1xx	CUN4BCPR_SrcSub_Chk
xxxx xx1x	CUN4BCPR_Bidi_Context
xxxx xxx1	CUN4BCPR_Bidi_ImpAlg

CUN4BCPR_Sub_Action

Specifies the action to take when a source character is encountered which is not convertible to the TO-CCSID.

- **0:** Indicates that the conversion is to be terminated with an error.
- **1:** Indicates that the substitution character is to be put in the target buffer and the conversion is to be continued.

CUN4BCPR_Inv_Handle

Specifies what has to be done when the UCCE handle is invalid.

- **0:** Indicates that the conversion is to be terminated with return code CUN_RC_WARN and reason code CUN_RS_INV_HANDLE_SET or CUN_RS_INV_HANDLE_NOSET.
- **1:** Indicates that the conversion is to be done with a new handle created by the conversion services and put into CUN4BCPR_Conv_Handle. This is done only if no SET UNI or SETUNI command is running. If the SET UNI command is still

running, the conversion will be terminated with return code CUN_RC_WARN and reason code CUN_RS_INV_HANDLE_SET.

CUN4BCPR_No_Opt_Buf_Fill

Specifies whether the target buffer is to be filled to a maximum for indirect conversion. This bit enables the caller to choose between fast execution without an optimally filled target buffer, and slower execution, but with a target buffer optimally filled.

- **0:** Indicates that the target buffer is to be filled to a maximum, taking additional steps into account. The benefit is that the target buffer is always filled with as many characters as possible, although processing time may be slow.
- **1:** Indicates that the target buffer is not filled to a maximum, which may decrease processing time. However, the number of characters that fit into the target buffer is only estimated once. Therefore, characters may be left in the source buffer, although the corresponding target characters would fit into the target buffer.

CUN4BCPR_Mal_Action

Specifies the action to take when a source character is malformed on the source CCSID. Note this action only occurs when CUN4BCPR_Sub_Action is 1.

- **0:** Indicates that the substitution character is to be put in the target buffer, and the conversion is to be continued when a malformed character is found.
- **1:** Indicates that the conversion is to be terminated with return code CUN_RC_WARN and reason code CUN_RS_MAL_CHAR_ACT_TERM, when a malformed character is found.

CUN4BCPR_RL_Sub_Action

Specifies what has to be done when "R" or "L" techniques are specified in the conversion call when a substitution character is converted.

- **0:** Indicates that CUN4BCPR_Sub_Act will work normally.
- **1:** Indicates that CUN4BCPR_Sub_Act will be overridden to 0 and no substitution bit (CUN4BCPR_Substitution) will be flagged.

CUN4BCPR_SrcSub_Chk

Specifies whether the service will consider source substitution chars as substitution or not.

- **0:** Indicates that the substitution character was placed in the target buffer when one or more malformed, invalid or substitution character were found in the source. In addition, the CUN4BCPR_Substitution flag, part of CUN4BCPR_Flag2, is turned on.
- **1:** Indicates that when a substitution character belonging to the FROM-CCSID is found in the source, a substitution character is placed in the target buffer, but the CUN4BCPR_Substitution flag is not turned on.

Note: This action only takes place when CUN4BCPR_Sub_Action is 1. In addition, it is highly recommended that exploiters of

this bit, notify their customers to rebuild their images, to avoid a degradation in performance.

CUN4BCPR_Bidi_Context

Specifies the context of the text to be transformed with the bidi service if technique B was specified.

- **0**: Indicates the context is Left to Right (LTR).
- **1**: Indicates the context is Right to Left (RTL).

CUN4BCPR_Bidi_ImpAlg

Specifies the algorithm to be used if technique B was specified.

- **0**: Indicates the basic algorithm will be used.
- **1**: Indicates the implicit algorithm will be used.

For more information, see Chapter 8, "Bidi transformation," on page 155.

CUN4BCPR_Subcodepage - set by caller initially, then set by conversion service

Used for conversions with CCSIDs that have a "state-dependent" encoding scheme (such as EBCDIC MBCS). For each new source string, on the first call to the character conversion service, CUN4BCPR_Subcodepage should be set to zero. Thus the converter will start with the default subcodepage(s). When the conversion service returns, CUN4BCPR_Subcodepage is updated to reflect the subcode page number used when converting the last source character. For subsequent calls to the character conversion service (partial string processing of long source strings), CUN4BCPR_Subcodepage must be used unchanged as returned from the previous call. Thus the next piece of source will start with the correct subcode page.

CUN4BCPR_Subcodepage is made up of two halfbytes. The first halfbyte can be referenced by the name CUN4BCPR_Source_SCP_State. The second halfbyte can be referenced by the name CUN4BCPR_Target_SCP_State.

CUN4BCPR_Source_SCP_State - set by caller initially, then set by conversion service

Reflects the FROM-CCSID's subcode page used for the last converted character. Specifically, CUN4BCPR_Source_SCP_State is set to:

0	To denote that a 'non-state' dependent' encoding scheme was used.
1	To denote that the last character converted came from an SBCS EBCDIC table.
2	To denote that the last character converted came from a DBCS EBCDIC table.
3	To denote that the last character converted came from a TBCS EBCDIC table.
4	To denote that the last character converted came from a QBCS EBCDIC table.
5	To denote that the last character converted came from an SBCS ASCII table.

- 6 To denote that the last character converted came from a DBCS ASCII table.
- 7 To denote that the last character converted came from a TBCS ASCII table.
- 8 To denote that the last character converted came from a QBCS ASCII table.

An easy way to get the value of this halfbyte is to 'AND' CUN4BCPR_Subcodepage with 'F0'.

CUN4BCPR_Source_SCP_State - set by caller initially, then set by conversion service

Reflects the TO-CCSID's subcode page used for the last converted character. Specifically, CUN4BCPR_Target_SCP_State is set to:

- 0 To denote that a 'non-state dependent' encoding scheme was used.
- 1 To denote that the last character converted came from an SBCS EBCDIC table.
- 2 To denote that the last character converted came from a DBCS EBCDIC table.
- 3 To denote that the last character converted came from a TBCS EBCDIC table.
- 4 To denote that the last character converted came from a QBCS EBCDIC table.
- 5 To denote that the last character converted came from an SBCS ASCII table.
- 6 To denote that the last character converted came from a DBCS ASCII table.
- 7 To denote that the last character converted came from a TBCS ASCII table.
- 8 To denote that the last character converted came from a QBCS ASCII table.

An easy way to get the value of this halfbyte is to 'AND' CUN4BCPR_Subcodepage with '0F'.

For example, when converting from MBCS to Unicode (UCS-2 or UTF-8) or any non-MBCS CCSID, CUN4BCPR_Source_SCP_State will be set. When converting from Unicode (UCS-2 or UTF-8) or any non-MBCS CCSID to MBCS, CUN4BCPR_Target_SCP_State will be set. When converting from any MBCS CCSID to another MBCS CCSID, both CUN4BCPR_Source_SCP_State and CUN4BCPR_Target_SCP_State will be set.

CUN4BCPR_Designator - set by conversion service

The parameter CUN4BCPR_Designator is used for conversions from and to ISO2022 encodings that use designator sequence. It specifies the active designator sequence in which the conversion is to begin. When the service returns, CUN4BCPR_Designator is updated as appropriate to reflect designator sequence active at the completion of the conversion.

For conversions to ISO2022-KR, which use only one designator, the sequence value means:

- **0:** The designator sequence was not yet inserted.
- **1:** The designator sequence was already inserted.

CUN4BCPR_Flag2 - set by service

Bit position	Name
1xxx xxxx	CUN4BCPR_Substitution
x1xx xxxx	CUN4BCPR_Mal_Found
xx1x xxxx	CUN4BCPR_Page_Fix
xxx1 xxxx	CUN4BCPR ETF3E_Behavior_Status

CUN4BCPR_Substitution

Indicates to the caller whether the conversion service has converted a character into the conversion table's substitution character.

Note: This bit has to be reset by the caller.

- **0:** Indicates that the conversion service did not substitute.
- **1:** Indicates that the conversion service converted at least one character into the conversion table's substitution character (or the service was already called with bit set to 1) .

CUN4BCPR_Mal_Found

Indicates to the caller whether the conversion service has encountered a malformed character in the source buffer.

- **0:** Indicates that the conversion service did not find a malformed character in the source buffer.
- **1:** Indicates that the conversion found at least one malformed character in the source buffer (or the service was already called with bit set to 1).

CUN4BCPR_Page_Fix

Indicates in case the conversion is not loaded whether it will be loaded on a Page fixed basis or not.

- **0:** Indicates Conversion will not be loaded on Page Fix.
- **1:** Indicates Conversion will be loaded on Page Fix.

Note:

- This bit has to be reset by the caller.
- CUN4BCPR_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUN4BCPR ETF3E_Behavior_Status

Indicates when CUN4BCPR ETF3E_Behavior is ON, whether the ETF3 hardware enhancement is in use for conversions from 1200 to 1208 and vice versa.

Note: When conversion are not requested from 1200 to 1208 and vice versa, the contents of this flag is not meaningful.

- **0:** Indicates ETF3 hardware enhancement is used. This is the default set.
- **1:** Indicates ETF3 hardware enhancement is not installed.

CUN4BCPR_RC_RS

Specifies a structure that can be used to access CUN4BCPR_Return_Code and CUN4BCPR_Reason_Code as one unit.

CUN4BCPR_Return_Code - set by service

Specifies the return code.

CUN4BCPR_Reason_Code - set by service

Specifies the reason code.

CUN4BCPR_Flag3 - set by caller

Bit position	Name
1xxx xxxx	CUN4BCPR ETF3E_Behavior

CUN4BCPR ETF3E_Behavior

Specify whether to exploit the ETF3 hardware enhancement for conversions from 1200 to 1208 and vice versa.

Note: To make this flag meaningful, the parameter area version field CUN4BCPR_Version must be defined as CUN4BCPR_Version2, otherwise this flag will be ignored.

- **0:** Do not exploit ETF3 hardware enhancement. 0 is the default.
- **1:** Exploit ETF3 hardware enhancement.

Handling a target buffer overflow

If the target buffer is too small, the conversion services will convert as many characters as will fit into the target buffer. When the service returns with the appropriate reason code for that situation, the source and target buffer pointers point to the byte following the last successfully converted source character (respectively inserted target character). Additionally, the source buffer length is updated to the number of bytes left unconverted in the source buffer and the target buffer length is updated to the number of bytes not yet consumed in the target buffer.

There are two ways in which a caller can respond to reason code CUN_RS_TRG_EXH (target buffer exhausted):

1. Redo the conversion with a large enough target buffer:

Repeat the conversion with a target buffer large enough to hold at least the maximum possible amount of target string bytes. To accomplish the necessary 'worst case' calculation, the caller has to take into account the number of source bytes to be converted and the nature of the CCSIDs involved (in terms of minimum possible source character width, maximum possible target character width, and possible shift-in/shift-out character sequences, or sub table switch control bytes). Such a 'worst case size' target buffer will prevent the occurrence of the reason code CUN_RS_TRG_EXH (target buffer exhausted).

The following table lists the minimum and maximum character widths of the different encoding schemes:

Table 8. Minimum and maximum character widths of the different encoding schemes

Encoding scheme	ESID	Minimum Character Width	Maximum Character Width	Rationale
SBCS	x1xx	1	1	pure single byte

Table 8. Minimum and maximum character widths of the different encoding schemes (continued)

Encoding scheme	ESID	Minimum Character Width	Maximum Character Width	Rationale
DBCS and UCS-2	x2xx	2	2	pure double byte
UTF-8	7807	1	4	UTF-8 uses 1 to 4 bytes to encode Unicode characters
PC MBCS	2300 to 3300	1	2	PC MBCS encodings always use one SBCS and one DBCS code page
EUC MBCS	4403	1	2 - 4	EUC encodings use at least one SBCS and at least one DBCS sub code page. If more than two sub code pages are used, shift characters are inserted for characters of the third and fourth sub code page. Then the maximum width is $2 + 1 = 3$. Some EUC encodings use TBCS (triple byte) code pages as the third sub code page (this case is not yet supported). Then the maximum width is $3 + 1 = 4$.
EBCDIC MBCS	1301	1	3	EBCDIC MBCS encodings always use one SBCS and one DBCS sub code page. Because switching between them is done with shift characters the maximum width is $2 + 1 = 3$.
ISO2022 MBCS JP and ISO2022 MBCS JP-1	5404	1	5 - 6	ISO2022 MBCS JP encodings always use at least one SBCS and at least one DBCS sub code page. Most ISO2022-JP encodings use an escape sequence of 4 characters for at least one of the DBCS sub code pages. Thus we get $2 + 4 = 6$. In one case the escape sequence is only 3 characters long. Then we get $2 + 3 = 5$.
ISO2022 MBCS KR	5409	1	6 - 7	ISO2022 MBCS KR encodings always use one or two SBCS sub code pages or one SCBS sub code page and one DBCS sub code page. Furthermore they use one designator sequence of length 4 before the first occurrence of a character of sub code page 2 and shift characters to switch between the sub code pages. Thus we get: $(1 \text{ or } 2) + 4 + 1 = (6 \text{ or } 7)$.
PC Data for GB 18030	2A00	1	4	S-ch PC Data mixed for GB 18030.
QBCS	2900	4	4	S-ch 4 bytes part PC Data for GB 18030 (Fixed UCS2 Subset).

2. Do the conversion piece by piece:

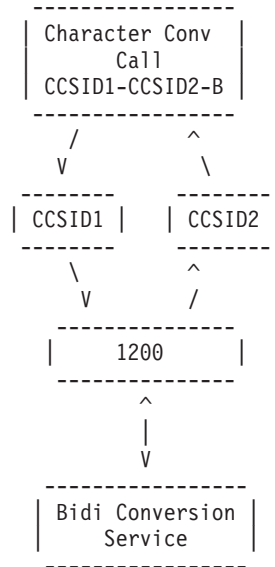
Save the target buffer characters already converted. Provide a new target buffer and call the conversion service again without modifying CUNBCPRM_Src_Buf_Len and CUNBCPRM_Src_Buf_Ptr to make sure that the conversion continues where it has been interrupted. This follow-on step may have to be repeated several times until all source bytes are converted. The

completion of the conversion is indicated by return code CUN_RC_OK (Return code=0). Concatenate the individual conversion results to form the complete converted string.

Character conversion service and the new B technique

As a feature on z/OS, V1R8 introduced Bidi Transformation services and, in addition, the provided stub routines to call on this service (CUNLBIDI and CUN4LBID). z/OS Unicode services offer this service with the character conversion service. Therefore, a new technique "B" is added exclusively for Bidi.

Character conversion service will sense when "B" technique is invoked and it will follow the path:



1. Start Unicode Character Conversion Service with technique "B".
2. If FROM-CCSID is not 1200 or other Unicode versions, convert the FROM-CCSID to CCSID 1200.
3. Call BIDI service.
4. If TO-CCSID is not 1200 or other Unicode versions, convert the BIDI output to TO-CCSID.
5. Return to the caller.

Sample programs

Sample programs for character conversion are provided in SYS1.SAMPLIB:

- CUNSCSMC for C
- CUNSCSMA for HLASM

End of Programming Interface information

Chapter 5. Case conversion

Programming Interface information

This chapter describes the programming required for the case conversion services.

Case conversion is also referred to as 'conversion to upper or lower case'. The case conversion services are called using a stub routine named **CUNLASE** for AMODE (31) and **CUN4LASE** for AMODE (64). It converts the case in a string of text characters.

Unicode case conversion is described in "Unicode Technical Report #21: Case Mappings" which is available at <http://www.unicode.org/>. Case conversion rules are summarized in the two tables **UnicodeData.txt** and **SpecialCasing.txt** which are available from the same web site.

To activate case conversion, specify the CASE control statement in the input data set for the image generator (job CUNMIUTL). For detailed information, see "Creating a conversion image" on page 9 and "Case conversion" on page 20.

The case conversion environment can also be dynamically activated when a conversion request is performed and the requested conversion has not been previously loaded.

Calling the case conversion services

This is a general description of how the case conversion services have to be called.

The 31 bit caller has to provide:

- Source buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Target buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Dynamic data area pointer (31 bit pointer), ALET (4 byte), and length (8 byte)
- Conversion type (or case conversion handle in subsequent calls)
 - Simple casing to upper or to lower
 - Locale independent special casing to upper or to lower
 - Locale dependent special casing to upper or to lower
- Flags

The 64-bit caller has to provide:

- Source buffer pointer (64 bit pointer), ALET (4 byte), and length (8 byte)
- Target buffer pointer (64 bit pointer), ALET (4 byte), and length (8 byte)
- Dynamic data area pointer (64 bit pointer), ALET (4 byte), and length (8 byte)
- Conversion type (or case conversion handle in subsequent calls)
 - Simple casing to upper or to lower
 - Locale independent special casing to upper or to lower
 - Locale dependent special casing to upper or to lower
- Flags

Note: A dynamic data area (DDA) must always be specified. The required length is defined by constant CUNBAPRM_DDA_Req for AMODE (31) and CUN4BAPR_DDA_Req for AMODE (64).

Case conversion

When the service returns, it replaces the source and target buffer pointers and lengths. Thus the caller can see how many bytes were converted and how much of the target buffer is filled up. Return codes and reason codes notify when a target buffer overflow was detected or any other critical case happened.

The conversion type is given initially. A call always returns a case conversion handle which is a fast path for the conversion services to the case conversion table and its properties. In subsequent calls, IBM recommends that you provide the case conversion handle. If the caller wants to request the case conversion handle without converting any data, it can be done by specifying a source buffer length of 0.

The caller can put the conversion data in any dataspace. To allow the service to access the data, an ALET must be specified. An ALET of 0 indicates that the data is in the primary address space.

Restrictions for the calling environment

Table 9. Restrictions while calling the case conversion services

Property	Restriction
Authorization	Problem state or supervisor state, and any PSW key
Dispatchable unit mode	Task or SRB
Cross memory mode	Any PASN, any HASN, any SASN
Amode	31-bit and 64-bit
ASC mode	Called in primary mode but exploiting AR mode
Interrupt status	Enabled for I/O and external interrupts
locks	May be held by the caller, but is not required to hold any
Control parameters	Must be in the primary address space
Recovery environment	Provided exclusively by the caller of the conversion services

Using the C interface

This is the call syntax in C for calling the stub routine **CUNLASE** (case conversion). The mapping of the parameter area supplied by the header file `cunhc.h` is listed in “Mapping of parameters in C” on page 61. A sample program, `CUNSASMC`, is provided in `SYS1.SAMPLIB`.

```
#include<cunhc.h>
#define SLEN 1000
#define TLEN 4096
.....
unsigned char Sourcebuffer [SLEN ];
unsigned char Targetbuffer [TLEN ];
unsigned char DDA [CUNBAPRM_DDA_REQ ];

CUNBAPRM myparm ={CUNBAPRM_DEFAULT};
myparm.Src_Buf_Ptr=Sourcebuffer;
myparm.Targ_Buf_Ptr=Targetbuffer;
myparm.Targ_Buf_Len=TLEN;
myparm.Src_Buf_Len=SLEN;
myparm.DDA_Buf_Ptr=DDA;
myparm.DDA_Buf_Length=CUNBAPRM_DDA_REQ;
Myparm.Conv_Type=CUNBAPRM_TO_UPPER;
```

```
CUNLASE ( & myparm );
if((myparm.Return_Code !=CUN_RC_OK).....
```

Mapping of parameters in C

A C header file is supplied (cunhc.h) which contains the function prototypes for the case conversion services. The following structure is used in the interface to the case conversion service.

31-bit mapping

```
typedef struct tagCUNBAPRM {
long      Version;           /* Structure version number */
long      Length;           /* Length of structure */
long      Res1;             /* Reserved */
void *    Src_Buf_Ptr;      /* Pointer to Source */
unsigned long Src_Buf_ALET;  /* ALET of source buffer */
unsigned long Src_Buf_Len;   /* Length of source data */
long      Res2;             /* Reserved */
void *    Targ_Buf_Ptr;     /* Pointer to Target */
unsigned long Targ_Buf_ALET; /* ALET of target buffer */
unsigned long Targ_Buf_Len;  /* Length of target buffer */
char      Conv_Handle[64];   /* conversion handle */
unsigned char Conv_Type;     /* conversion type */
char      Res3[3];          /* Reserved */
char      Locale[32];       /* LOCALE */
long      Res4;             /* Reserved */
void *    DDA_Buf_Ptr;      /* Pointer to dynamic data area */
unsigned long DDA_Buf_ALET;  /* ALET of DDA */
unsigned long DDA_Buf_Len;   /* Length of DDA */
    struct {
        int      Inv_Handle      : 1, /* Invalid handle action: */
                                     /* 0 = Terminate with error */
                                     /* 1 = Get new handle and */
        Not_Last_Buf : 1, /* Buffer contains last */
                                     /* Source Character */
                                     /* 0 = Src_Buffer is last/only */
                                     /* Buffer of complete src data */
                                     /* 1 = Another buffer follows */
        Page_Fix      : 1, /* Page fixing: */
                                     /* 0=System storage */
                                     /* 1=Page Fixing. */
                                     : 5; /* FLAG Byte 1 set by caller */
    } Flag1;
    struct {
        int      Locale_Support : 1, /* Locale support: */
                                     /* When RC/RS <> 8/4 meaning: */
                                     /* 0 = Locale supported */
                                     /* When RC/RS = 8/4 meaning: */
                                     /* 1 = Invalid Locale name */
                                     /* When RC/RS <> 8/4 meaning: */
                                     /* 1 = Locale Not supported */
                                     /* (locale name is valid) */
                                     : 7; /* Padding */
    } Flag2; /* Flag2 - set by the service */
    unsigned char Res5[2]; /* Reserved */
    long      Return_Code;
    long      Reason_Code;
    unsigned char Res6[3]; /* Reserved */
    unsigned char UniVersion; /* Unicode Data version */
} CUNBAPRM;
```

Case conversion

Note: C constants for the parameter area are defined in the header file cunhc.h.

64-bit mapping

```
typedef struct tagCUN4BAPR {
    unsigned int    Version;           /* Structure version number */
    unsigned int    Length;            /* Length of structure */
    void *          Src_Buf_Ptr;       /* Pointer to Source */
    unsigned int    Src_Buf_ALET;      /* ALET of source buffer */
    unsigned int    Res1;              /* Reserved */
    unsigned long   Src_Buf_Len;       /* Length of source data */
    void *          Targ_Buf_Ptr;      /* Pointer to Target */
    unsigned int    Targ_Buf_ALET;     /* ALET of target buffer */
    unsigned int    Res2;              /* Reserved */
    unsigned long   Targ_Buf_Len;      /* Length of target buffer */
    char            Conv_Handle[64];   /* conversion handle */
    char            Conv_Type;         /* conversion type */
    char            Res3[7];
    char            Locale[32];        /* LOCALE */
    void *          DDA_Buf_Ptr;       /* Pointer to dynamic data area */
    unsigned int    DDA_Buf_ALET;      /* ALET of DDA */
    unsigned int    DDA_Buf_Len;       /* Length of DDA */
    struct {
        int         Inv_Handle        : 1, /* Invalid handle action: */
                                           /* 0 = Terminate with error */
                                           /* 1 = Get new handle and */
                                           /* Buffer contains last */
                                           /* Source Character */
                                           /* 0 = Src_Buffer is last/only */
                                           /* Buffer of complete src data */
                                           /* 1 = Another buffer follows */
        int         Page_Fix          : 1, /* Page fixing: */
                                           /* 0=System storage */
                                           /* 1=Page Fixing. */
        int         : 5; /* FLAG Byte 1 set by caller */
    } Flag1;
    struct {
        int         Locale_Support    : 1, /* Locale support: */
                                           /* When RC/RS <> 8/4 meaning: */
                                           /* 0 = Locale supported */
                                           /* When RC/RS = 8/4 meaning: */
                                           /* 1 = Invalid Locale name */
                                           /* When RC/RS <> 8/4 meaning: */
                                           /* 1 = Locale Not supported */
                                           /* (locale name is valid) */
        int         : 7; /* Padding */
    } Flag2; /* Flag2 - set by the service */
    unsigned char   Res5[2];           /* Reserved */
    int             Return_Code;
    int             Reason_Code;
    unsigned char   Res6[3];          /* Reserved */
    unsigned char   UniVersion;       /* Unicode Data version */
} CUN4BAPR;
```

Using the HLASM interface

This is the call syntax in HLASM for calling the stub routine **CUNLASE** (case conversion for 31-bit callers) and **CUN4LASE** (case conversion for 64-bit callers). A sample program, CUNSASMA, is provided in SYS1.SAMPLIB.

For AMODE (31)

```
-----1-----2-----3-----4-----5-----6-----7--
      GETMAIN .....      Obtain storage for parameter area
```



```

*                                     in primary address space.
LR    R4,R1                          Save parameter area address
USING CUNBAPRM,R4                     Make parameter area addressable
XC    CUNBAPRM,CUNBAPRM               Init PARAMETER AREA TO BINARY 0
LA    R15,CUNBAPRM_VER                Get Version
ST    R15,CUNBAPRM_VERSION            Store to parameter area
LA    R15,CUNBAPRM_LEN                Initialize Length
ST    R15,CUNBAPRM_LENGTH             Move to parameter area
LA    R0,CUNBAPRM_TO_UPPER            Get conversion type
STC   R0,CUNBAPRM_CONV_TYPE           Store to parameter area

*
* Supply source buffer pointer, length and ALET.
* Supply target buffer pointer, length and ALET.
* Supply DDA buffer pointer, length and ALET.
* Note: A DDA is always required. The required DDA length is
* defined by constant CUNBAPRM_DDA_REQ.
*
* Fill all required fields of the parameter area.
CALL  CUNLASE,((R4))                 Call stub routine with CUNBAPRM
*                                     address as argument.
CUNBAIDF DSECT=YES                   Provide Mappings (CUNBAPRM, return and
*                                     reason codes, constants for version
*                                     and length).

```

For AMODE (64)

```

-----1-----2-----3-----4-----5-----6-----7--
GETMAIN .....                      Obtain storage for parameter area
*                                     in primary address space
LR    R4,R1                          Save parameter area address
USING CUN4BAPR,R4                     Make parameter area addressable
XC    CUN4BAPR,CUN4BAPR               Init PARAMETER AREA TO BINARY 0
LA    R15,CUN4BAPR_VER                Get Version
ST    R15,CUN4BAPR_VERSION            Version Store to parameter area
LA    R15,CUN4BAPR_LEN                Initialize Length
ST    R15,CUN4BAPR_LENGTH             Move to parameter area
LA    R0,CUN4BAPR_TO_UPPER            Get conversion type
ST    R0,CUN4BAPR_CONV_TYPE           Store to parameter area

*
* Supply source buffer pointer, length and ALET.
* Supply target buffer pointer, length and ALET.
* Supply DDA buffer pointer, length and ALET.
* Note: A DDA is always required. The required DDA length is
* defined by constant CUN4BAPR_DDA_REQ.
* Set flags
*
CALL  CUN4LASE,((R4))                 Call stub routine with CUN4BAPR
*                                     address as argument.
CUN4BAID DSECT=YES                     Provide Mappings (CUN4BAPR, return and
*                                     reason codes, constants for version
*                                     and length).

```

Mapping of parameters for AMODE (31)

The mapping of the parameter areas is supplied by the interface definition file CUNBAIDF. This file is shipped in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that may be necessary.

Table 10. Mapping of parameters in HLASM for case conversion AMODE (31)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	168	DWORD	CUNBAPRM	Parameter Area

Case conversion

Table 10. Mapping of parameters in HLASM for case conversion AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	UNSIGNED	4		CUNBAPRM_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUNBAPRM_Length	Parameter area Length
8	(8)	CHARACTER	4		*	Reserved for 64 bit
12	(C)	ADDRESS	4		CUNBAPRM_Src_Buf_Ptr	Source buffer pointer
16	(10)	UNSIGNED	4		CUNBAPRM_Src_Buf_ALET	Source buffer ALET
20	(14)	UNSIGNED	4		CUNBAPRM_Src_Buf_Len	Source buffer length
24	(18)	CHARACTER	4		*	Reserved for 64 bit
28	(1C)	ADDRESS	4		CUNBAPRM_Targ_Buf_Ptr	Target buffer pointer
32	(20)	UNSIGNED	4		CUNBAPRM_Targ_Buf_ALET	Target buffer ALET
36	(24)	UNSIGNED	4		CUNBAPRM_Targ_Buf_Len	Target buffer length
40	(28)	CHARACTER	64	DWORD	CUNBAPRM_Conv_Handle	Conversion handle
104	(68)	UNSIGNED	1		CUNBAPRM_Conv_Type	Conversion Type
105	(69)	CHARACTER	3		*	Reserved
108	(6C)	CHARACTER	32		CUNBAPRM_Locale	Locale info
140	(8C)	CHARACTER	4		*	Reserved for 64 bit
144	(90)	ADDRESS	4		CUNBAPRM_DDA_Buf_Ptr	Dynamic data area pointer
148	(94)	UNSIGNED	4		CUNBAPRM_DDA_Buf_ALET	Dynamic data area ALET
152	(98)	UNSIGNED	4		CUNBAPRM_DDA_Buf_Len	Dynamic data area length as defined by constant CUNBAPRM_DDA_Req.
156	(9C)	BITSTRING	1		CUNBAPRM_Flag1	FLAG Byte 1 set by caller
		1... ..			CUNBAPRM_Inv_Handle	Invalid handle action: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONTINUE.
		.1... ..			CUNBAPRM_Not_Last_Buf	Buffer contains last src char: 0=Src_Buffer is last or only buffer of complete src data. 1=Another buffer follows.
		..1.			CUNBAPRM_Page_Fix	Page fixing: 0=System storage 1=Page Fixing
157	(9D)	UNSIGNED	1		CUNBAPRM_Flag2	FLAG Byte 2 (Set by caller)
		1... ..	..		CUNBAPRM_Locale_Support	Locale support: When RC/RS <> 8/4 meaning: 0 = Locale supported When RC/RS = 8/4 meaning: 1 = Invalid Locale name When RC/RS <> 8/4 meaning: 1 = Locale Not supported (locale name is valid)
158	(9E)	CHARACTER	2		*	Reserved

Table 10. Mapping of parameters in HLASM for case conversion AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
160	(A0)	CHARACTER	8	WORD	CUNBAPRM_RC_RS	Return/reason code
		UNSIGNED	4		CUNBAPRM_Return_Code	Return code
		UNSIGNED	4		CUNBAPRM_Reason_Code	Reason code
168	(A8)	CHARACTER	3		*	Reserved
171	(AB)	CHARACTER	8		CUNBAPRM_UniVersion	Unicode Data Version
179	(B3)		0	WORD	CUNBAPRM_End	End of CUNBAPRM

Description of parameters in area CUNBAPRM

This description applies to C and HLASM.

CUNBAPRM_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUNLASE using the constant CUNBAPRM_Version that is supplied by the interface definition file CUNBAIDF.

As of V1R9 and later releases, new parameter area is supported. If CUNBAPRM_Version is set to CUNBAPRM_Ver2, new CASE service features might be exploited:

- Exploit “Tittle Case” features (See CUNBAPRM_Conv_Type)
- Use specific Unicode character version (See CUNBAPRM_UniVersion)

CUNBAPRM_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUNLASE using the constant CUNBAPRM_length which is supplied by the interface definition file CUNBAIDF.

CUNBAPRM_Src_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of a string of text characters which are to be converted. The string has the length specified in the CUNBAPRM_Src_Buf_Len parameter. At the completion of the conversion, CUNBAPRM_Src_Buf_Ptr will be updated to point just past the last character that was successfully converted, and CUNBAPRM_Src_Buf_Len will be updated to reflect the number of bytes left unconverted. If all bytes are converted, CUNBAPRM_Src_Buf_Len will be zero.

Note: Source buffer pointed by CUNBAPRM_Src_Buf_Ptr must contain UTF-16 BE characters format only. Otherwise, CASE Conversion Service will cause unpredictable results.

CUNBAPRM_Src_Buf_ALET - set by caller

Specifies the ALET to be used, if the source buffer addressed by CUNBAPRM_Src_Buf_Ptr resides in a different address or data space.

CUNBAPRM_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUNBAPRM_Src_Buf_Ptr, to be converted. The source buffer length may be zero. In this case, nothing is converted but the CUNBAPRM_Conv_Handle is returned. This may be used to request a handle without converting.

Case conversion

CUNBAPRM_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage where the converted text string will be stored. At the completion of the conversion, CUNBAPRM_Targ_Buf_Ptr will point just past the last character stored, and CUNBAPRM_Targ_Buf_Len will be updated to indicate the number of bytes not yet consumed in the buffer.

CUNBAPRM_Targ_Buf_ALET - set by caller

Specifies the ALET to be used, if the target buffer addressed by CUNBAPRM_Targ_Buf_Ptr resides in a different address or data space.

CUNBAPRM_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUNBAPRM_Targ_Buf_Ptr.

CUNBAPRM_Conv_Handle - set by conversion service

Specifies the handle to the case conversion tables. If a handle is present, it will be used, otherwise the CUNBAPRM_Conv_Type and CUNBAPRM_UniVersion (if provided) parameters are used and a case conversion handle is returned in CUNBAPRM_Conv_Handle. Subsequent calls to stub routine CUNLASE, requesting the same conversion, will be faster because the handle is used and CUNBAPRM_Conv_Type does not need to be recomputed.

Note: For the first call to stub routine CUNLASE, CUNBAPRM_Conv_Handle must be set to binary zero X'00'.

CUNBAPRM_Conv_Type - set by caller

Specifies the conversion direction as defined by the following constants:

CUNBAPRM_To_Upper	Converts to upper case, includes simple casing only
CUNBAPRM_To_Lower	Converts to lower case, includes simple casing only
CUNBAPRM_To_Upper_S	Converts to upper case, includes locale independent special casing
CUNBAPRM_To_Lower_S	Converts to lower case, includes locale independent special casing
CUNBAPRM_To_Upper_L	Converts to upper case, includes locale dependent and independent special casing
CUNBAPRM_To_Lower_L	Converts to lower case, includes locale dependent and independent special casing
CUNBAPRM_To_Title	Converts to title case, includes simple casing only
CUNBAPRM_To_Title_S	Converts to title case, includes locale independent special casing
CUNBAPRM_To_Title_L	Converts to title case, includes locale dependent and independent special casing

Conversion types CUNBAPRM_To_Title, CUNBAPRM_To_Title_S and CUNBAPRM_To_Title_L can be used only if CUNBAPRM_Version is set to CUNBAPRM_Ver2 and if CUNBAPRM_UniVersion is not set to one of the following:

- CUNBAPRM_NONE
- CUNBAPRM_UNI300

Other valid Unicode data versions can use those case conversion types.

CUNBAPRM_Locale - set by caller

Specifies the locale information to be used when the locale dependent special casing is specified (Conv_Type = CUNBAPRM_TO_UPPER_L, CUNBAPRM_TO_LOWER_L or CUNBAPRM_To_Title_L). The locale can use the form *LL_CC* where

LL is a two-letter language code (for example **tr** for Turkish).

CC is a two-letter country code (for example **TR** for Turkey).

Note: LL and CC are not case sensitive. All input will be folded to uppercase. However, when specifying locale names in lower case, a non-Katakana EBCDIC CCSID must be used.

If the locale name is not specified, only locale independent special casing will be performed.

The following table lists the locales currently supported:

Table 11. Case Conversion Service supported locales - AMODE(31)

Unicode Version	Locales Supported	Locale Description
UNI300	tr_tr	Turkish / Turkey
UNI301	tr_tr lt_lt	Turkish / Turkey Lithuanian / Lithuania
UNI320	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan
UNI401	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan
UNI410	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan
UNI510	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan

If the locale name specified is not supported, the case conversion service will return with RC=CUN_RC_USER_ERROR, RS=CUN_RS_CASE_NOT_SUPP.

CUNBAPRM_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion service is using internally as dynamic data area.

Note: CUNBAPRM_DDA_Buf_Ptr must be double-word boundary.

CUNBAPRM_DDA_Buf_ALET - set by caller

Specifies the ALET to be used if the dynamic data area addressed by CUNBAPRM_DDA_Ptr resides in a different address or data space.

CUNBAPRM_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUNBAPRM_DDA_Ptr.

Note: If CUNBAPRM_Version is set to CUNBAPRM_Ver2, you must set CUNBAPRM_DDA_Buf_Len to CUNBAPRM_DDA_Req_Ver2.

CUNBAPRM_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUNBAPRM_Inv_Handle
x1xx xxxx	CUNBAPRM_Not_Last_Buf
xx1x xxxx	CUNBAPRM_Page_Fix

CUNBAPRM_Inv_Handle

Specifies the action to be taken when the case conversion handle is invalid.

- **0:** Indicates that the conversion is to be terminated with an error.
- **1:** Indicates that the conversion is to be done with a new handle created by the conversion service and put into CUNBAPRM_Conv_Handle.

CUNBAPRM_Not_Last_Buf

Specifies whether the source buffer contains the last or only part of the complete source data, or whether the next call to the case converter will supply a subsequent part of the source data.

- **0:** Indicates that the source buffer contains the last or only part of the source data.
- **1:** Indicates that another buffer with more source characters will be supplied with the subsequent call to case conversion.

CUNBAPRM_Page_Fix

Specifies the action to be taken for the page fixing:

- **0:** Indicates use of system storage management.
- **1:** Indicates use of page fixing.

Note: CUNBAPRM_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUNBAPRM_Flag2 - set by conversion service

Bit position	Name
1xxx xxxx	CUNBAPRM_Locale_Support

CUNBAPRM_Locale_Support

Indicates to the caller whether the locale provided by CUNBAPRM_Locale was supported, not supported or invalid.

- **Locale Supported:** CUNBAPRM_Locale content matches one of the locale names (See CUNBAPRM_Locale for a list of supported locales).
- **Locale Invalid:** CUNBAPRM_Locale content does not match any of the locale names from Appendix G, "Case service and locale valid names," on page 479.
- **Locale NOT supported:** CUNBAPRM_Locale_Support content matches one of the locale names for Case service support list (See Appendix G, "Case service and locale valid names," on page 479).

Terms	CUNBAPRM_Locale_Support Value	Description
Supported	0	When Return/Reason Code is <i>not</i> set to CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP This means that the locale is supported. For any other Return/Reason Code, the flag might be set, but it is not related with a locale handling error.
Invalid	1	Return/Reason Code is set to CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP This means that the Locales name is not valid, and Case Services returns to the caller.
NOT supported	1	When Return/Reason Code is not set to CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP This means that locale is <i>not</i> supported; however, conversion continues. For any other Return/Reason Code, the flag might be set, but it is not related with a locale handling error.

Note: Result of this CUNBAPRM_Locale_Support flag is meaningful when callers request a Case Locale type only, that is, CUNBAPRM_To_Upper_L or CUNBAPRM_To_Lower_L. Any other case type (for example, CUNBAPRM_To_Upper, CUNBAPRM_To_Lower, and so on) in combination with this flag is *not* meaningful.

CUNBAPRM_RC_RS

Specifies a structure that can be used to access CUNBAPRM_Return_Code and CUNBAPRM_Reason_Code as one unit.

CUNBAPRM_Return_Code - set by conversion service

Specifies the return code.

CUNBAPRM_Reason_Code - set by conversion service

Specifies the reason code.

CUNBAPRM_UniVersion - set by caller

Specifies the Unicode data version. This field is meaningful for the case conversion service, only if CUNBAPRM_Version is set to CUNBAPRM_Ver2. Valid values are:

- CUNBAPRM_NONE (DEFAULT), 3.0.0. Unicode data version is requested.
- CUNBAPRM_UNI300, 3.0.0 Unicode data version is requested.
- CUNBAPRM_UNI301, 3.0.1 Unicode data version is requested.
- CUNBAPRM_UNI320, 3.2.0 Unicode data version is requested.
- CUNBAPRM_UNI401, 4.0.1 Unicode data version is requested.
- CUNBAPRM_UNI410, 4.1.0 Unicode data version is requested.
- CUNBAPRM_UNI500, 5.0.0 Unicode data version is requested.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas is supplied by the interface definition file CUN4BAID. This file is shipped in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that may be necessary.

Table 12. Mapping of parameters in HLASM for case conversion AMODE (64)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	184	DWORD	CUN4BAPR	Parameter Area
0	(0)	UNSIGNED	4		CUN4BAPR_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUN4BAPR_Length	Parameter area Length
8	(8)	CHARACTER	8		CUN4BAPR_Src_Buf_Ptr	Source buffer pointer
16	(10)	ADDRESS	4		CUN4BAPR_Src_Buf_ALET	Source buffer ALET
20	(14)	UNSIGNED	4		*	Reserved
24	(18)	UNSIGNED	8		CUN4BAPR_Src_Buf_Len	Source buffer length
32	(20)	CHARACTER	8		CUN4BAPR_Targ_Buf_Ptr	Target buffer pointer
40	(28)	ADDRESS	4		CUN4BAPR_Targ_Buf_ALET	Target buffer ALET
44	(2C)	UNSIGNED	4		*	Reserved for 64 bit
48	(30)	UNSIGNED	8		CUN4BAPR_Targ_Buf_Len	Target buffer length
56	(38)	CHARACTER	64	DWORD	CUN4BAPR_Conv_Handle	Conversion handle
120	(78)	UNSIGNED	1		CUN4BAPR_Conv_Type	Conversion Type
121	(79)	CHARACTER	3		*	Reserved
124	(7C)	CHARACTER	32		CUN4BAPR_Locale	Language locale used for case conversion
156	(9C)	CHARACTER	4		*	Reserved for 64 bit
160	(A0)	ADDRESS	4		CUN4BAPR_DDA_Buf_Ptr	Dynamic data area pointer
164	(A4)	UNSIGNED	4		CUN4BAPR_DDA_Buf_ALET	Dynamic data area ALET
168	(A8)	UNSIGNED	4		CUN4BAPR_DDA_Buf_Len	Dynamic data area length as defined by constant CUN4BAPR_DDA_Req.
172	(AC)	BITSTRING	1		CUN4BAPR_Flag1	FLAG Byte 1 set by caller
		1... ..			CUN4BAPR_Inv_Handle	Invalid handle action: 0=TERMINATE WITH ERROR. 1=GET NEW HANDLE AND CONT.
		.1... ..			CUN4BAPR_Not_Last_Buf	Buffer contains last src char 0=SRC_BUFFER IS LAST OR ONLY PART OF COMPLETE SRC DATA. 1=ANOTHER BUFFER FOLLOWS.
		..1.			CUN4BAPR_Page_Fix	Page fixing: 0=System storage 1=Page Fixing
173	(AD)	UNSIGNED	1		CUN4BAPR_Flag2	FLAG Byte 2 (Set by caller)

Table 12. Mapping of parameters in HLASM for case conversion AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
		1...	..		CUN4BAPR_Locale_Support	Locale support: When RC/RS <> 8/4 meaning: 0 = Locale supported When RC/RS = 8/4 meaning: 1 = Invalid Locale name When RC/RS <> 8/4 meaning: 1 = Locale Not supported (locale name is valid)
174	(AE)	CHARACTER	2		*	Reserved
176	(B0)	CHARACTER	8	WORD	CUN4BAPR_RC_RS	Return/reason code
		UNSIGNED	4		CUN4BAPR_Return_Code	Return code
		UNSIGNED	4		CUN4BAPR_Reason_Code	Reason code
184	(B8)	CHARACTER	3		*	Reserved
187	(BB)	CHARACTER	8		CUN4BAPR_UniVersion	Unicode Data Version
195	(C3)		0	WORD	CUN4BAPR_End	End of CUN4BAPR

Description of parameters in area CUN4BAPR

This description applies to C and HLASM.

CUN4BAPR_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUNLASE using the constant CUN4BAPR_Version that is supplied by the interface definition file CUN4BAID.

As of V1R9 and later releases, the new parameter area is supported. If CUN4BAPR_Version is set to CUN4BAPR_Ver2, new CASE service features might be exploited:

- Exploit “Tittle Case” features (See CUN4BAPR_Conv_Type)
- Use specific Unicode character version (See CUN4BAPR_UniVersion)

CUN4BAPR_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUNLASE using the constant CUN4BAPR_length which is supplied by the interface definition file CUN4BAID.

CUN4BAPR_Src_Buf_Ptr - set by caller, updated by service

Specifies the first eight bytes of address of a string of text characters which are to be converted. The string has the length specified in the CUN4BAPR_Src_Buf_Len parameter. At the completion of the conversion, CUN4BAPR_Src_Buf_Ptr will be updated to point just past the last character that was successfully converted, and CUN4BAPR_Src_Buf_Len will be updated to reflect the number of bytes left unconverted. If all bytes are converted, CUN4BAPR_Src_Buf_Len will be zero.

Note: Source buffer pointed by CUN4BAPR_Src_Buf_Ptr must contain UTF-16 BE characters format only. Otherwise, CASE Conversion Service will cause unpredictable results.

Parameters in area CUN4BAPR

CUN4BAPR_Src_Buf_ALET - set by caller

Specifies the ALET to be used if the source buffer addressed by CUN4BAPR_Src_Buf_Ptr resides in a different address or data space.

CUN4BAPR_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUN4BAPR_Src_Buf_Ptr, to be converted. The source buffer length may be zero. In this case, nothing is converted but the CUN4BAPR_Conv_Handle is returned. This may be used to request a handle without converting.

CUN4BAPR_Targ_Buf_Ptr - set by caller

Specifies the first eight bytes of address of an area of storage where the converted text string will be stored. At the completion of the conversion, CUN4BAPR_Targ_Buf_Ptr will point just past the last character stored, and CUN4BAPR_Targ_Buf_Len will be updated to indicate the number of bytes not yet consumed in the buffer.

CUN4BAPR_Targ_Buf_ALET - set by caller

Specifies the ALET to be used if the target buffer addressed by CUN4BAPR_Targ_Buf_Ptr resides in a different address or data space.

CUN4BAPR_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUN4BAPR_Targ_Buf_Ptr.

CUN4BAPR_Conv_Handle - set by conversion service

Specifies the handle to the case conversion tables. If a handle is present, it will be used, otherwise the CUN4BAPR_Conv_Type and CUN4BAPR_UniVersion (if provided) parameters are used and a case conversion handle is returned in CUN4BAPR_Conv_Handle. Subsequent calls to stub routine CUN4LASE, requesting the same conversion, will be faster because then the handle is used and CUN4BAPR_Conv_Type does not need to be recomputed..

Note: For the first call to stub routine CUNLASE, CUN4BAPR_Conv_Handle must be set to binary zero X'00'.

CUN4BAPR_Conv_Type - set by caller

Specifies the conversion direction as defined by the following constants:

CUN4BAPR_To_Upper	Converts to upper case, includes simple casing only
CUN4BAPR_To_Lower	Converts to lower case, includes simple casing only
CUN4BAPR_To_Upper_S	Converts to upper case, includes locale independent special casing
CUN4BAPR_To_Lower_S	Converts to lower case, includes locale independent special casing
CUN4BAPR_To_Upper_L	Converts to upper case, includes locale dependent and independent special casing
CUN4BAPR_To_Lower_L	Converts to lower case, includes locale dependent and independent special casing
CUN4BAPR_To_Title	Converts to title case, includes simple casing only
CUN4BAPR_To_Title_S	Converts to title case, includes locale independent special casing

CUN4BAPR_To_Title_L	Converts to title case, includes locale dependent and independent special casing

Conversion types CUN4BAPR_To_Title, CUN4BAPR_To_Title_S and CUN4BAPR_To_Title_L can be used only if CUN4BAPR_Version is set to CUN4BAPR_Ver2 and if CUN4BAPR_UniVersion is not set to one of the following:

- CUN4BAPR_NONE
- CUN4BAPR_UNI300

Other valid Unicode data versions can use those case conversion types.

CUN4BAPR_Locale - set by caller

Specifies the locale information to be used when the locale dependent special casing is specified (Conv_Type = CUN4BAPR_TO_UPPER_L, CUN4BAPR_TO_LOWER_L or CUN4BAPR_To_Title_L). The locale can use the form *LL_CC* where

LL is a two-letter language code (for example **tr** for Turkish).

CC is a two-letter country code (for example **TR** for Turkey).

Note: LL and CC are not case sensitive. All input will be folded to uppercase. However, when specifying locale names in lower case, a non-Katakana EBCDIC CCSID must be used.

If the locale name is not specified, only locale independent special casing will be performed.

The following table lists the locales currently supported:

Table 13. Case Conversion Service supported locales - AMODE(64)

Unicode Version	Locales Supported	Locale Description
UNI300	tr_tr	Turkish / Turkey
UNI301	tr_tr lt_lt	Turkish / Turkey Lithuanian / Lithuania
UNI320	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan
UNI401	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan
UNI410	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan
UNI510	tr_tr lt_lt az_az	Turkish / Turkey Lithuanian / Lithuania Azeri / Azerbaijan

If the locale name specified is not supported, the case conversion service will return with RC=CUN_RC_USER_ERROR, RS=CUN_RS_CASE_NOT_SUPP.

CUN4BAPR_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion service is using internally as dynamic data area.

Note: CUN4BAPR_DDA_Buf_Ptr must be double-word boundary.

CUN4BAPR_DDA_Buf_ALET - set by caller

Specifies the ALET to be used if the dynamic data area addressed by CUN4BAPR_DDA_Ptr resides in a different address or data space.

Parameters in area CUN4BAPR

CUN4BAPR_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUN4BAPR_DDA_Ptr.

Note: If CUN4BAPR_Version is set to CUN4BAPR_Ver2, you must set CUN4BAPR_DDA_Buf_Len to CUN4BAPR_DDA_Req_Ver2.

CUN4BAPR_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUN4BAPR_Inv_Handle
x1xx xxxx	CUN4BAPR_Not_Last_Buf
xx1x xxxx	CUN4BAPR_Page_Fix

CUN4BAPR_Inv_Handle

Specifies the action to be taken when the case conversion handle is invalid.

- **0:** Indicates that the conversion is to be terminated with an error.
- **1:** Indicates that the conversion is to be done with a new handle created by the conversion service and put into CUN4BAPR_Conv_Handle.

CUN4BAPR_Not_Last_Buf

Specifies whether the source buffer contains the last or only part of the complete source data, or whether the next call to the case converter will supply a subsequent part of the source data.

- **0:** Indicates that the source buffer contains the last or only part of the source data.
- **1:** Indicates that another buffer with more source characters will be supplied with the subsequent call to case conversion.

CUN4BAPR_Page_Fix

Specifies the action to be taken for the page fixing:

- **0:** Indicates use of system storage management.
- **1:** Indicates use of page fixing.

Note: CUN4BAPR_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUN4BAPR_Flag2 - set by conversion service

Bit position	Name
1xxx xxxx	CUN4BAPR_Locale_Support

CUN4BAPR_Locale_Support

Indicates to the caller whether the locale provided by CUN4BAPR_Locale was supported, not supported, or invalid

- **Locale Supported:** CUN4BAPR_Locale content matches one of the locale names (See CUN4BAPR_Locale for a list of supported locales).
- **Locale Invalid:** CUN4BAPR_Locale content does not match any of the locale names from the Appendix G, "Case service and locale valid names," on page 479 section .

- **Locale NOT supported:** CUN4BAPR_Locale content matches one of the locale names for the case service support (See Appendix G, “Case service and locale valid names,” on page 479).

Terms	CUN4BAPR_Locale_Support Value	Description
Supported	0	When Return/Reason Code is not set to CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP This means that the locale is supported. For any other Return/Reason Code, the flag might be set, but it is not related with a locale handling error.
Invalid	1	When Return/Reason Code is set to CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP This means that the locale name is <i>not</i> valid, and Case Services returns to the caller.
NOT supported	1	When Return/Reason Code is <i>not</i> set to CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP This means that the locale is <i>not</i> supported; however, conversion continues. For any other Return/Reason Code, the flag might be set but it is not related with a locale handling error.

Note: Result of this CUN4BAPR_Locale_Support flag is meaningful when callers request a Case Locale type only, that is, CUN4BAPR_To_Upper_L or CUN4BAPR_To_Lower_L. Any other case type (that is, CUN4BAPR_To_Upper, CUN4BAPR_To_Lower, and so on) in combination with this flag is *not* meaningful.

CUN4BAPR_RC_RS

Specifies a structure that can be used to access CUN4BAPR_Return_Code and CUN4BAPR_Reason_Code as one unit.

CUN4BAPR_Return_Code - set by conversion service

Specifies the return code.

CUN4BAPR_Reason_Code - set by conversion service

Specifies the reason code.

CUN4BAPR_UniVersion - set by caller

Specifies the Unicode data version. This field is meaningful for the case conversion service, only if CUN4BAPR_Version is set to CUN4BAPR_Ver2. Valid values are:

- CUN4BAPR_NONE (DEFAULT), 3.0.0. Unicode data version is requested.
- CUN4BAPR_UNI300, 3.0.0 Unicode data version is requested.
- CUN4BAPR_UNI301, 3.0.1 Unicode data version is requested.
- CUN4BAPR_UNI320, 3.2.0 Unicode data version is requested.

Parameters in area CUN4BAPR

- CUN4BAPR_UNI401, 4.0.1 Unicode data version is requested.
- CUN4BAPR_UNI410, 4.1.0 Unicode data version is requested.
- CUN4BAPR_UNI500, 5.0.0 Unicode data version is requested.

Sample programs

Sample programs for case conversion are provided in SYS1.SAMPLIB:

31-bit samples

- CUNSASMC for C
- CUNSASMA for HLASM

64-bit samples

- CUN4A02C for C
- CUN4A01A for HLASM

_____ **End of Programming Interface information** _____

Chapter 6. Normalization

Programming Interface information

This chapter describes the programming required for the Normalization services.

Normalization is also referred to as decomposition or composition. The normalization service is called using a stub routine named **CUNLNORM** for AMODE (31) and **CUN4LNOR** for AMODE (64). Normalization allows the decomposition or composition of a Unicode input string. Normalization is described in "Unicode Technical Report #15: Unicode Normalization Forms", which is available at <http://www.unicode.org/unicode/reports/tr15>.

Normalization rules are based on the following Unicode versions:

Table 14. Unicode version table

Unicode version		URL
UNI301	UnicodeData-3.0.1.txt	http://www.unicode.org/Public/3.0-Update1/UnicodeData-3.0.1.txt
	CompositionExclusions-2.txt	http://www.unicode.org/Public/3.0-Update1/CompositionExclusions-2.txt
UNI320	UnicodeData-3.2.0.txt	http://www.unicode.org/Public/3.2-Update/UnicodeData-3.2.0.txt
	CompositionExclusions-3.2.0.txt	http://www.unicode.org/Public/3.2-Update/CompositionExclusions-3.2.0.txt
UNI401	UnicodeData-4.0.1.txt	http://www.unicode.org/Public/4.0-Update1/UnicodeData-4.0.1.txt
	CompositionExclusions-4.0.0.txt	http://www.unicode.org/Public/4.0-Update/CompositionExclusions-4.0.0.txt
UNI410	UnicodeData.txt	http://www.unicode.org/Public/4.1.0/ucd/UnicodeData.txt
	CompositionExclusions.txt	http://www.unicode.org/Public/4.1.0/ucd/CompositionExclusions.txt

Normalization can be activated by specifying the NORMALIZE control statement in the input data set for the image generator. For detailed information see "Creating a conversion image" on page 9 and "Normalization conversion" on page 21. The normalization environment can also be dynamically activated when a conversion request is performed and the requested conversion has not been previously loaded.

Calling the normalization service

This is a general description of how the normalization services have to be called.

The 31 bit caller has to provide:

- Source buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Target buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Work buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Normalization form (NFC, NFD, NFKD or NFKC)
- Dynamic data area pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Flags
- Unicode Version

Normalization

The 64-bit caller has to provide:

- Source buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Work buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Normalization form (NFC, NFD, NFKD or NFKC)
- Dynamic data area pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Flags
- Unicode Version

Note: A dynamic data area (DDA) must always be specified. The required length is defined by constant CUNBNPRM_DDA_Req for AMODE (31) and CUN4BNPR_DDA_Req for AMODE (64).

On a successful return from the normalization service, the data area pointed by the target buffer pointer as long as the target, source buffer pointers and lengths are updated. The caller can see how many bytes were normalized and how much of the target buffer is filled up. In case of any error, return codes and reason codes are updated with necessary information.

Handling a work buffer overflow

For the normalization service, it is strongly recommended that the work buffer be at least the same size as the target buffer. If not, an error could occur, such as RC=CUN_RC_USER_ERR and RS=CUN_RS_WRK_BUF_SMALL. In this case the normalization service returns to the caller.

Restrictions for the calling environment

Table 15. Restrictions while calling the normalization service

Property	Restriction
Authorization	Problem state or supervisor state, and any PSW key
Dispatchable unit mode	Task or SRB
Cross memory mode	Any PASN, any HASN, any SASN
AMODE	31-bit and 64-bit
ASC mode	Called in primary mode but exploiting AR mode
Interrupt status	Enabled for I/O and external interrupts
Locks	May be held by the caller, but is not required to hold any
Control parameters	Must be in the primary address space
Recovery environment	Provided exclusively by the caller of the normalization service

Using the C interface

This is the call syntax in C for calling the stub routine **CUNLNORM** (normalization). The mapping of the parameter area supplied by the header file cunhc.h is listed in “Mapping of parameters in C” on page 61. A sample program, CUNSNSMC, is provided in SYS1.SAMPLIB.

```
#include<cunhc.h>
#define SLEN 10
```



```

#define WLEN 40
#define TLEN 40
.....
unsigned char Sourcebuffer[SLEN]=
{'\x00','\x41','\x00','\x41','\x00','\xC0','\x00','\x41','\x00','\x41'};
unsigned char Workbuffer [WLEN ];
unsigned char Targetbuffer [TLEN ];

unsigned char DDA [CUNBNPRM_DDA_REQ ];
CUNBNPRM myparm ={CUNBNPRM_DEFAULT};

myparm.Src_Buf_Ptr=Sourcebuffer;
myparm.Wrk_Buf_Ptr=Workbuffer;
myparm.Targ_Buf_Ptr=Targetbuffer;

myparm.Targ_Buf_Len=TLEN;
myparm.Wrk_Buf_Len=WLEN;
myparm.Src_Buf_Len=SLEN;

myparm.DDA_Buf_Ptr=DDA;
myparm.DDA_Buf_Length=CUNBNPRM_DDA_REQ;
myparm.Norm_Type=CUNBNPRM_D;
CUNLNORM ( & myparm );
if((myparm.Return_Code !=CUN_RC_OK).....

```

Mapping of parameters in C

A C header file is supplied (cunhc.h) that contains the function prototypes for the normalization service. The following structure is used in the interface to the normalization service.

31-bit mapping

```

typedef struct tagCUNBNPRM {
    long          Version;           /* Structure version number */
    long          Length;            /* Length of structure */
    long          Res1;              /* Reserved */
    void *        Src_Buf_Ptr;       /* Pointer to Source */
    unsigned long Src_Buf_ALET;      /* ALET of source buffer */
    unsigned long Src_Buf_Len;       /* Length of source data */
    long          Res2;              /* Reserved */
    void *        Targ_Buf_Ptr;      /* Pointer to Target */
    unsigned long Targ_Buf_ALET;     /* ALET of target buffer */
    unsigned long Targ_Buf_Len;      /* Length of target buffer */
    char          Norm_Handle[64];   /* Normalization handle */
    unsigned char Norm_Type;         /* normalization type */
    unsigned char Res3[7];           /* Reserved */
    long          Res4;              /* Reserved */
    void *        Wrk_Buf_Ptr;       /* Pointer to work buffer */
    unsigned long Wrk_Buf_ALET;      /* ALET of work buffer */
    unsigned long Wrk_Buf_Len;       /* Length of work buffer */
    long          Res5;              /* Reserved */
    void *        DDA_Buf_Ptr;       /* Pointer to dynamic data area */
    unsigned long DDA_Buf_ALET;      /* ALET of DDA */
    unsigned long DDA_Buf_Len;       /* Length of DDA */
    struct {
        int      Inv_Handle          : 1, /* Invalid handle action: */
                                           /* 0 = Terminate with error */
                                           /* 1 = Get new handle and */
        int      Page_Fix            : 1, /* Page Fixing: */
                                           /* 0 = System storage */
                                           /* 1 = Page Fixing */
        : 6;
    };
};

```

Normalization

```
    } Flag1;                                /* FLAG Byte 1 set by caller */
    unsigned char Res6[3];                   /* Reserved */
    long          Return_Code;               /* Return code */
    long          Reason_Code;              /* Reason code */
    unsigned char Res7[3];                   /* Reserved */
    unsigned char UniVersion;                /* Unicode Data version for
                                           /* Normalization tables */

} CUNBNPRM;
```

64-bit mapping

```
typedef struct tagCUN4BNPR {
    unsigned int  Version;                   /* Structure version number */
    unsigned int  Length;                   /* Length of structure */
    void *        Src_Buf_Ptr;               /* Pointer to Source */
    unsigned int  Src_Buf_ALET;              /* ALET of source buffer */
    unsigned int  Res1;                     /* Reserved */
    unsigned long Src_Buf_Len;               /* Length of source data */
    void *        Targ_Buf_Ptr;              /* Pointer to Target */
    unsigned int  Targ_Buf_ALET;             /* ALET of target buffer */
    unsigned int  Res2;                     /* Reserved */
    unsigned long Targ_Buf_Len;              /* Length of target buffer */
    char          Norm_Handle[64];           /* Normalization handle */
    unsigned char Norm_Type;                 /* normalization type */
    unsigned char Res3[7];                   /* Reserved */
    void *        Wrk_Buf_Ptr;               /* Pointer to work buffer */
    unsigned int  Wrk_Buf_ALET;              /* ALET of work buffer */
    unsigned int  Res4;                     /* Reserved */
    unsigned long Wrk_Buf_Len;               /* Length of work buffer */
    void *        DDA_Buf_Ptr;               /* Pointer to dynamic data area */
                                           /* */
    unsigned int  DDA_Buf_ALET;              /* ALET of DDA */
    unsigned int  DDA_Buf_Len;              /* Length of DDA */
    struct {
        int      Inv_Handle      : 1, /* Invalid handle action: */
                                           /* 0 = Terminate with error */
                                           /* 1 = Get new handle and */
        int      Page_Fix       : 1, /* Page Fixing: */
                                           /* 0 = System storage */
                                           /* 1 = Page Fixing */
        int      : 6;
    } Flag1;                                /* FLAG Byte 1 set by caller */
    unsigned char Res6[3];                   /* Reserved */
    unsigned int  Return_Code;               /* Return code */
    unsigned int  Reason_Code;              /* Reason code */
    unsigned char Res7[3];                   /* Reserved */
    unsigned char UniVersion;                /* Unicode Data version for
                                           /* Normalization tables */

} CUN4BNPR;
```

Note: C constants for the parameter area are defined in the header file cunhc.h.

Using the HLASM interface

This is the call syntax in HLASM for calling the stub routine **CUNLNORM** (normalization for AMODE (31)) and **CUN4LNOR** (normalization for AMODE (64)). A sample program, CUNSNSMA, is provided in SYS1.SAMPLIB.

For AMODE (31)

-----1-----2-----3-----4-----5-----6-----7--

```

      GETMAIN .....      Obtain storage for parameter area
*                               in primary address space.
      LR   R4,R1           Save parameter area address
      USING CUNBNPRM,R4    Make parameter area addressable
      XC   CUNBNPRM,CUNBNPRM  Init PARAMETER AREA TO BINARY 0
      LA   R15,CUNBNPRM_VER  Get Version
      ST   R15,CUNBNPRM_VERSION Store to parameter area
      LA   R15,CUNBNPRM_LEN  Initialize Length
      ST   R15,CUNBNPRM_LENGTH Move to parameter area
      LA   R0,CUNBNPRM_D     Get normalization type
      STC  R0,CUNBNPRM_NORM_TYPE Store to parameter area

*
* Supply source buffer pointer, length and ALET.
* Supply work buffer pointer, length and ALET.
* Supply target buffer pointer, length and ALET.
*
* Supply DDA buffer pointer, length and ALET.
* Note: A DDA is always required. The required DDA length is
* defined by constant CUNBNPRM_DDA_REQ.
*
* Fill all required fields of the parameter area.
* CALL CUNLNORM,((R4)) Call stub routine with CUNBNPRM
* address as argument.
* CUNBNIDF DSECT=YES      Provide Mappings (CUNBNPRM, return and
* reason codes, constants for version
* and length).

```

For AMODE (64)

```

-----1-----2-----3-----4-----5-----6-----7--

      GETMAIN .....      Obtain storage for parameter area
*                               in primary address space
      LR   R4,R1           Save parameter area address
      USING CUN4BNPR,R4    Make parameter area addressable
      XC   CUN4BNPR,CUN4BNPR  Init PARAMETER AREA TO BINARY 0
      LA   R15,CUN4BNPR_VER  Get Version
      ST   R15,CUN4BNPR_VERSION Version Store to parameter area
      LA   R15,CUN4BNPR_LEN  Initialize Length
      ST   R15,CUN4BNPR_LENGTH Move to parameter area
      LA   R0,CUN4BNPR_D     Get normalization type
      STC  R0,CUN4BNPR_NORM_TYPE Store to parameter area

*
* Supply source buffer pointer, length and ALET.
* Supply work buffer pointer, length and ALET.
* Supply target buffer pointer, length and ALET.
*
* Supply DDA buffer pointer, length and ALET.
* Note: A DDA is always required. The required DDA length is
* defined by constant CUN4BNPR_DDA_REQ.
*
* Fill all required fields of the parameter area.
* CALL CUN4LNOR,((R4)) Call stub routine with CUN4BNPR
* address as argument.
* CUN4BNID DSECT=YES      Provide Mappings (CUN4BNPR, return and
* reason codes, constants for version
* and length).

```

Mapping of parameters for AMODE (31)

The mapping of the parameter areas is supplied by the interface definition file CUNBNIDF. This file is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

Normalization

Table 16. Mapping of parameters in HLASM for normalization AMODE (31)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	160	DWORD	CUNBNPRM	Parameter Area
0	(0)	UNSIGNED	4		CUNBNPRM_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUNBNPRM_Length	Parameter area Length
8	(8)	CHARACTER	4		*	Reserved for 64 bit
12	(C)	ADDRESS	4		CUNBNPRM_Src_Buf_Ptr	Source buffer pointer
16	(10)	UNSIGNED	4		CUNBNPRM_Src_Buf_ALET	Source buffer ALET
20	(14)	UNSIGNED	4		CUNBNPRM_Src_Buf_Len	Source buffer length
24	(18)	CHARACTER	4		*	Reserved for 64 bit
28	(1C)	ADDRESS	4		CUNBNPRM_Targ_Buf_Ptr	Target buffer pointer
32	(20)	UNSIGNED	4		CUNBNPRM_Targ_Buf_ALET	Target buffer ALET
36	(24)	UNSIGNED	4		CUNBNPRM_Targ_Buf_Len	Target buffer length
40	(28)	CHARACTER	64	DWORD	CUNBNPRM_Norm_Handle	Normalization handle
104	(68)	UNSIGNED	1		CUNBNPRM_Norm_Type	Normalization Type
105	(69)	CHARACTER	7		*	Reserved
112	(70)	CHARACTER	4		*	Reserved for 64 bit
116	(74)	ADDRESS	4		CUNBNPRM_Wrk_Buf_Ptr	Work buffer pointer
120	(78)	UNSIGNED	4		CUNBNPRM_Wrk_Buf_ALET	Work buffer ALET
124	(7C)	UNSIGNED	4		CUNBNPRM_Wrk_Buf_Len	Work buffer length
128	(80)	CHARACTER	4		*	Reserved for 64 bit
132	(84)	ADDRESS	4		CUNBNPRM_DDA_Buf_Ptr	Dynamic data area pointer
136	(88)	UNSIGNED	4		CUNBNPRM_DDA_Buf_ALET	Dynamic data area ALET
140	(8C)	UNSIGNED	4		CUNBNPRM_DDA_Buf_Len	Dynamic data area length as defined by constant CUNBNPRM_DDA_Req.
144	(90)	BITSTRING	1		CUNBNPRM_Flag1	FLAG Byte 1 set by caller
		1... ..			CUNBNPRM_Inv_Handle	Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONTINUE.
		.1.. ..			CUNBNPRM_Page_Fix	Page Fixing: 0=System storage management (default). 1=Page fixing.
		..11 1111			*	Reserved
145	(91)	CHARACTER	3		*	Reserved
148	(94)	CHARACTER	8	WORD	CUNBNPRM_RC_RS	Return/reason code
		UNSIGNED	4		CUNBNPRM_Return_Code	Return code
		UNSIGNED	4		CUNBNPRM_Reason_Code	Reason code
156	(9C)	CHARACTER	3		*	Reserved

Table 16. Mapping of parameters in HLASM for normalization AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
159	(9F)	CHARACTER	1		CUNBNPRM_UniVersion	normalization Unicode data version
160	(A0)		0	WORD	CUNBNPRM_End	End of CUNBNPRM

Description of parameters in area CUNBNPRM

This description applies to C and HLASM.

CUNBNPRM_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUNLNORM using the constant CUNBNPRM_Ver which is supplied by the interface definition file CUNBNIDF.

Also, if callers want to exploit new normalization data versions, this field must be set with CUNBNPRM_Ver2, which is defined in CUNBNIDF. With this value, the normalization algorithm uses the normalization data version as specified in the new field CUNBNPRM_UniVersion. See CUNBNPRM_UniVersion parameter description for a list of valid values.

If **CUNBNPRM_Version** is set with CUN4BNPR_Ver, the contents of **CUNBNPRM_UniVersion** is not significant, and normalization data version 3.0.1 is assumed.

CUNBNPRM_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUNLNORM using the constant CUNBNPRM_Len which is supplied by the interface definition file CUNBNIDF.

CUNBNPRM_Src_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of a string of text characters. At the completion of the normalization, CUNBNPRM_Src_Buf_Ptr will be updated to point just past the last character that was successfully normalized. If all bytes are normalized, CUNBNPRM_Src_Buf_Len will be zero.

Note: Source buffer pointed by CUNBNPRM_Src_Buf_Ptr must contain UTF-16 BE characters format only. Otherwise, Normalization Service will cause unpredictable results.

CUNBNPRM_Src_Buf_ALET - set by caller

Specifies the ALET to be used to access the source buffer addressed by CUNBNPRM_Src_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUNBNPRM_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUNBNPRM_Src_Buf_Ptr, to be normalized. The source buffer length may be zero. In this case nothing is normalized, but the CUNBNPRM_Norm_Handle is returned. This may be used to request a handle without normalizing.

CUNBNPRM_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage where the normalized text string will be stored. At the completion of the normalization,

Normalization

CUNBNPRM_Targ_Buf_Ptr will point just past the last character stored, and CUNBNPRM_Targ_Buf_Len will be updated to indicate the number of bytes not yet consumed in the buffer.

CUNBNPRM_Targ_Buf_ALET - set by caller

Specifies the ALET to be used to access the target buffer addressed by CUNBNPRM_Targ_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUNBNPRM_Targ_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the target buffer addressed by CUNBNPRM_Targ_Buf_Ptr. It is strongly suggested this length be at least the same size as CUNBNPRM_Src_Buf_Len.

CUNBNPRM_Norm_Handle - set by caller, updated by service

CUNBNPRM_Norm_Handle specifies the handle to the normalization tables. If a handle is present, it will be used, otherwise the CUNBNPRM_Norm_Type and CUNBNPRM_UniVersion (if provided) parameters are used, and a normalization handle is returned in CUNBNPRM_Norm_Handle. Subsequent calls to stub routine CUNLNORM, requesting the same normalization, will be faster because then the handle is used and CUNBNPRM_Norm_Type does not need to be recomputed.

Note: For the first call to stub routine CUNLNORM, CUNBNPRM_Norm_Handle must be set to binary zero X'00'.

CUNBNPRM_Norm_Type - set by caller

Specifies the normalization type as defined by the following constants (defined in CUNBNIDF):

CUNBNPRM_D	Normalize to canonical decomposition
CUNBNPRM_C	Normalize to canonical composition
CUNBNPRM_KD	Normalize to compatibility decomposition
CUNBNPRM_KC	Normalize to compatibility composition

CUNBNPRM_Wrk_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of an area of storage that the normalization service can use to store intermediate results.

CUNBNPRM_Wrk_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffer addressed by CUNBNPRM_Wrk_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUNBNPRM_Wrk_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work buffer addressed by CUNBNPRM_Wrk_Buf_Ptr. It is strongly suggested this length be at least the same size as CUNBNPRM_Targ_Buf_Len

CUNBNPRM_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the normalization service is using internally as a dynamic data area.

Note: CUNBNPRM_DDA_Buf_Ptr must be double-word boundary.

CUNBNPRM_DDA_Buf_ALET - set by caller

Specifies the ALET to be used to access the dynamic data area addressed by CUNBNPRM_DDA_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUNBNPRM_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUNBNPRM_DDA_Buf_Ptr. The required length is defined by constant CUNBNPRM_DDA_Req.

CUNBNPRM_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUNBNPRM_Inv_Handle
x1xx xxxx	CUNBNPRM_Page_Fix

CUNBNPRM_Inv_Handle

Specifies the action to be taken when the normalization handle is invalid:

- **0:** Indicates that the normalization is to be terminated with an error.
- **1:** Indicates that the normalization is to be done with a new handle created by the normalization service and put into CUNBNPRM_Norm_Handle.

CUNBNPRM_Page_Fix

Specifies the action to be taken for the page fixing:

- **0:** Indicates use of system storage management (default).
- **1:** Indicates use of page fixing.

Note: CUNBNPRM_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUNBNPRM_Return_Code - set by service

Specifies the return code.

CUNBNPRM_Reason_Code - set by service

Specifies the reason code.

CUNBNPRM_UniVersion - set by caller

Specifies the normalization Unicode data version. This field is meaningful for the normalization algorithm and Unicode dynamic capabilities only if CUNBNPRM_Version is set to CUNBNPRM_Ver2. Valid values are:

- CUNBNPRM_NONE (DEFAULT), 3.0.1 Unicode data version is requested.
- CUNBNPRM_UNI301, 3.0.1 Unicode data version is requested.
- CUNBNPRM_UNI320, 3.2.0 Unicode data version is requested.
- CUNBNPRM_UNI401, 4.0.1 Unicode data version is requested.
- CUNBNPRM_UNI410, 4.1.0 Unicode data version is requested.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas is supplied by the interface definition file CUN4BNID. This file is shipped in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that may be necessary.

Normalization

Table 17. Mapping of parameters in HLASM for normalization AMODE (64)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	184	DWORD	CUN4BNPR	Parameter Area
0	(0)	UNSIGNED	4		CUN4BNPR_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUN4BNPR_Length	Parameter area Length
8	(8)	ADDRESS	8		CUN4BNPR_Src_Buf_Ptr	Source buffer pointer
16	(10)	ADDRESS	4		CUN4BNPR_Src_Buf_ALET	Source buffer ALET
20	(14)	UNSIGNED	4		*	Reserved
24	(18)	UNSIGNED	8		CUN4BNPR_Src_Buf_Len	Source buffer length
32	(20)	ADDRESS	8		CUN4BNPR_Targ_Buf_Ptr	Target buffer pointer
40	(28)	ADDRESS	4		CUN4BNPR_Targ_Buf_ALET	Target buffer ALET
44	(2C)	UNSIGNED	4		*	Reserved for 64 bit
48	(30)	UNSIGNED	8		CUN4BNPR_Targ_Buf_Len	Target buffer length
56	(38)	CHARACTER	64	DWORD	CUN4BNPR_Norm_Handle	Normalization handle
120	(78)	UNSIGNED	1		CUN4BNPR_Norm_Type	Normalization Type
121	(79)	CHARACTER	7		*	Reserved
128	(80)	ADDRESS	8		CUN4BNPR_Wrk_Buf_Ptr	Work buffer pointer
136	(88)	UNSIGNED	4		CUN4BNPR_Wrk_Buf_ALET	Work buffer ALET
140	(8C)	CHARACTER	4		*	Reserved
144	(90)	UNSIGNED	4		CUN4BNPR_Wrk_Buf_Len	Work buffer length
152	(98)	ADDRESS	8		CUN4BNPR_DDA_Buf_Ptr	Dynamic data area pointer
160	(A0)	UNSIGNED	4		CUN4BNPR_DDA_Buf_ALET	Dynamic data area ALET
164	(A4)	UNSIGNED	4		CUN4BNPR_DDA_Buf_Len	Dynamic data area length as defined by constant CUN4BNPR_DDA_Req.
168	(A8)	BITSTRING	1		CUN4BNPR_Flag1	FLAG Byte 1 set by caller
		1... ..			CUN4BNPR_Inv_Handle	Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONTINUE.
		.1..			CUN4BNPR_Page_Fix	Page Fixing: 0=System storage management (default). 1=Page fixing.
		..11 1111			*	Reserved
169	(A9)	CHARACTER	3		*	Reserved
172	(AC)	CHARACTER	8	WORD	CUN4BNPR_RC_RS	Return/reason code
		UNSIGNED	4		CUN4BNPR_Return_Code	Return code
		UNSIGNED	4		CUN4BNPR_Reason_Code	Reason code
180	(B4)	CHARACTER	3		*	Reserved
183	(B7)	CHARACTER	1		CUN4BNPR_UniVersion	normalization Unicode data version

Table 17. Mapping of parameters in HLASM for normalization AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
184	(B8)		0	WORD	CUN4BNPR_End	End of CUN4BNPR

Description of parameters in area CUN4BNPR

This description applies to C and HLASM.

CUN4BNPR_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUN4LNOR using the constant CUN4BNPR_Ver which is supplied by the interface definition file CUN4BNID.

Also, if callers want to exploit new normalization data versions, this field must be set with CUN4BNPR_Ver2, which is defined in CUN4BNID. With this value, normalization algorithm uses the normalization data version as specified in the new field CUN4BNPR_UniVersion . See CUN4BNPR_UniVersion parameter description for a list of valid values.

If **CUN4BNPR_Version** is set with CUN4BNPR_Ver, the contents of **CUN4BNPR_UniVersion** is not significant, and normalization data version 3.0.1 is assumed.

CUN4BNPR_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUN4LNOR using the constant CUN4BNPR_Len which is supplied by the interface definition file CUN4BNID.

CUN4BNPR_Src_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of a string of text characters. At the completion of the normalization, CUN4BNPR_Src_Buf_Ptr will be updated to point just past the last character that was successfully normalized. If all bytes are normalized, CUN4BNPR_Src_Buf_Len will be zero.

Note: Source buffer pointed by CUN4BNPR_Src_Buf_Ptr must contain UTF-16 BE characters format only. Otherwise, Normalization Service will cause unpredictable result.

CUN4BNPR_Src_Buf_ALET - set by caller

Specifies the ALET to be used to access the source buffer addressed by CUN4BNPR_Src_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BNPR_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUN4BNPR_Src_Buf_Ptr, to be normalized. The source buffer length may be zero. In this case nothing is normalized, but the CUN4BNPR_Norm_Handle is returned. This may be used to request a handle without normalizing.

CUN4BNPR_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage where the normalized text string will be stored. At the completion of the normalization,

Normalization

CUN4BNPR_Targ_Buf_Ptr will point just past the last character stored, and CUN4BNPR_Targ_Buf_Len will be updated to indicate the number of bytes not yet consumed in the buffer.

CUN4BNPR_Targ_Buf_ALET - set by caller

Specifies the ALET to be used to access the target buffer addressed by CUN4BNPR_Targ_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BNPR_Targ_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the target buffer addressed by CUN4BNPR_Targ_Buf_Ptr. It is strongly suggested this length be at least the same size as CUN4BNPR_Src_Buf_Len.

CUN4BNPR_Norm_Handle - set by caller, updated by service

Specifies the handle to the normalization tables. If a handle is present, it will be used, otherwise the CUN4BNPR_Norm_Type and CUN4BNPR_UniVersion (if provided) parameters are used, and a normalization handle is returned in CUN4BNPR_Norm_Handle. Subsequent calls to stub routine CUN4LNOR, requesting the same normalization, will be faster because then the handle is used and CUN4BNPR_Norm_Type does not need to be recomputed.

Note: For the first call to stub routine CUN4LNOR, CUN4BNPR_Norm_Handle must be set to binary zero X'00'.

CUN4BNPR_Norm_Type - set by caller

Specifies the normalization type as defined by the following constants (defined in CUNBNIDF):

CUN4BNPR_D	Normalize to canonical decomposition
CUN4BNPR_C	Normalize to canonical composition
CUN4BNPR_KD	Normalize to compatibility decomposition
CUN4BNPR_KC	Normalize to compatibility composition

CUN4BNPR_Wrk_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of an area of storage that the normalization service can use to store intermediate results.

CUN4BNPR_Wrk_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffer addressed by CUN4BNPR_Wrk_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BNPR_Wrk_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work buffer addressed by CUN4BNPR_Wrk_Buf_Ptr. It is strongly suggested this length be at least the same size as CUN4BNPR_Targ_Buf_Len

CUN4BNPR_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the normalization service is using internally as dynamic data area.

Note: CUN4BNPR_DDA_Buf_Ptr must be double-word boundary.

CUN4BNPR_DDA_Buf_ALET - set by caller

Specifies the ALET to be used to access the dynamic data area addressed by CUN4BNPR_DDA_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BNPR_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUN4BNPR_DDA_Buf_Ptr. The required length is defined by constant CUN4BNPR_DDA_Req.

CUN4BNPR_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUN4BNPR_Inv_Handle
x1xx xxxx	CUN4BNPR_Page_Fix

CUN4BNPR_Inv_Handle

Specifies the action to be taken when the normalization handle is invalid.

- **0:** Indicates that the normalization is to be terminated with an error.
- **1:** Indicates that the normalization is to be done with a new handle created by the normalization service and put into CUN4BNPR_Norm_Handle.

CUN4BNPR_Page_Fix

Specifies the action to be taken for the page fixing:

- **0:** Indicates use of system storage management (default).
- **1:** Indicates use of page fixing.

Note: CUN4BNPR_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUN4BNPR_Return_Code - set by service

Specifies the return code.

CUN4BNPR_Reason_Code - set by service

Specifies the reason code.

CUN4BNPR_UniVersion - set by caller

Specifies the normalization Unicode data version. Possible values are: This field is meaningful for the normalization algorithm and Unicode dynamic capabilities only if CUN4BNPR_Version is set to CUN4BNPR_Ver2. Valid values are:

- CUN4BNPR_NONE (DEFAULT), 3.0.1 Unicode data version is requested.
- CUN4BNPR_UNI301, 3.0.1 Unicode data version is requested.
- CUN4BNPR_UNI320, 3.2.0 Unicode data version is requested.
- CUN4BNPR_UNI401, 3.0.1 Unicode data version is requested.
- CUN4BNPR_UNI410, 4.1.0 Unicode data version is requested.

Sample programs

Sample programs for normalization are provided in SYS1.SAMPLIB:

31-bit samples

- CUNSNSMC for C
- CUNSNSMA for HLASM

Normalization

64-bit samples

- CUN4SNSA for C
- CUN4SNSC for HLASM

_____ **End of Programming Interface information** _____

Chapter 7. Collation

Programming Interface information

This chapter describes the programming required for the collation services.

The collation service provides a way for making culturally correct comparisons between two input Unicode strings according to the Unicode Services collation algorithm. It can also be used to generate a sort key for one or two Unicode strings. A sort key is a collection of weights which is optionally created in the collation process and is binary compared against another sort key to produce a compare result. Once a sort key is generated it can be kept and later used to do compares between other sort keys.

Since z/OS V1R8 and later, collation supports customization, which means that collation service might behave according to some specific collation rules. Collation rules might be specified using a Locale or by some User Collation Rules (UCR). The following are the new collation versions:

UCA400R1

This collation version supports Unicode standard character suite 4.0.0 and uses Normalization Service under 4.0.1 Unicode character suite.

UCA410

This collation version supports Unicode standard character suite 4.1.0 and uses Normalization Service under 4.1.0 Unicode character suite.

Current collation version (UCA301) covers 3.0.1 and does not support customization.

This z/OS Unicode implementation uses the instructions in the z/Architecture® Extended-Translation Facility 1 and 2 on models where those facilities are supported. The Extended-Translation Facility instructions can result in significant improvements in the performance of Unicode Services processing.

This z/OS collation implementation meets the specifications described in the Unicode Standard Versions 3.0.1, 4.0.0 and 4.1.0. For further information about the Unicode collation standard, refer to the Unicode Consortium technical report #10 (<http://www.Unicode.org/Unicode/reports/tr10>).

The collation service can be called through stub routine CUNLOCOL for AMODE (31) or CUN4LCOL for AMODE (64). To create a Unicode image with collation, the COLLATE control statement must be present in the image generator (job CUNMIUTL).

IMPORTANT: z/OS Unicode Collation Service requires the normalization services if a collation is called with parameter CUNBOPRM_Norm_Type, specifying a particular normalization form (see “Description of parameters in area CUNBOPRM” on page 113). In this case, the image generator requires the NORMALIZE statement be present also.

Collation version 4.0.0 requires normalization service 4.0.1 and collation version 4.1.0 requires normalization service 4.1.0, which are supported by z/OS V1R8 and later. See Chapter 6, “Normalization,” on page 77 for more information.

For detailed information, see “Creating a conversion image” on page 9 and “Collation conversion” on page 21.

Calling the collation service

This section describes how the z/OS support for Unicode collation service is called.

Collation works under two basic schemes — the binary comparison between two Unicode strings, and the generation of a sort key vector. Following is a description of how the service is called, followed by an explanation of the uses of the two types of calls.

Binary comparison

The 31-bit caller has to provide:

- Source1 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Source2 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Target1 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Target2 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Collation level
- Work1 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Work2 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Dynamic data area pointer (DDA) (31-bit pointer), ALET (4 byte), and length (8 byte)
- Flag1 (handle options)
- Collation mask options (sort key option=0)

The 64-bit caller has to provide:

- Source1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Source2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Collation level
- Work1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Work2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Dynamic data area pointer (DDA) (64 bit pointer), ALET (4 byte), and length (8 byte)
- Flag1 (handle options)
- Collation mask options (sort key option=0)

For new collation features (UCA400R1 and UCA410), there are two ways to set the APIs as part of Unicode Dynamic Capabilities:

1. Long Path. This way to perform Collation API settings has the intention to continue to use the existing collation settings “plus” the new ones
2. Short Path. This new way to set Collation API is a very simple and easy for all the collation features supported.

Another option is to use SETUNI or SET UNI=xx commands as part of an static initialization. For more information, see SETUNI command in *z/OS MVS System Commands*.

Long Path

The 31-bit caller has to provide:

- Set parameter area version2
- Source1 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Source2 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Target1 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Target2 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Collation level
- Work1 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Work2 buffer pointer (31-bit pointer), ALET (4 byte), and length (8 byte)
- Dynamic data area pointer (DDA) (31-bit pointer), ALET (4 byte), and length (8 byte)
- Flag1 (handle options)
- Collation mask options (sort key option=0)
- Case Options Flags
- Hiragana support
- Locale or User Collation Rules file + DSN + Vol

The 64-bit caller has to provide:

- Set parameter area version2
- Source1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Source2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Collation level
- Work1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Work2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Dynamic data area pointer (DDA) (64 bit pointer), ALET (4 byte), and length (8 byte)
- Flag1 (handle options)
- Collation mask options (sort key option=0)
- Case Options Flags
- Hiragana support
- Locale or User Collation Rules file + DSN + Vol

Short Path

The 31-bit caller has to provide:

- Set parameter area version2
- Source1 buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Source2 buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Target1 buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Target2 buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Work2 buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Dynamic data area pointer (DDA) (31 bit pointer), ALET (4 byte), and length (4 byte)

- Collation Keyword

The 64-bit caller has to provide:

- Set parameter area version2
- Source1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Source2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Work2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Dynamic data area pointer (DDA) (64 bit pointer), ALET (4 byte), and length (8 byte)
- Collation Keyword

Note: Short path settings has high priority over long path.

Sort key vector

How you generate the sort key vector depends on how you set the sourceX buffer length. For example, you can use any of the following input combinations:

- Source1
- Source2
- Source1 and source2

In the first two cases, you only need to provide the pointers for the applicable source, work, and target buffers. In case number three, you must provide pointers for both sets of buffers.

You always must provide the following, regardless of which of the above scenarios applies:

- Collation level
- Dynamic data area pointer (DDA), ALET, and length
- Flag1 (handle options)
- Collation mask options (sort key option=1)

Following is an explanation of the two types of calls to the collation service.

1. Binary comparison

This is the most common use of the collation service. Two Unicode strings are input by the caller to be compared (collated) in a culturally correct manner. Prior to collation, the caller must provide a desired collation level and optionally, the alternate weighting, and other options in the collation parameter area, to specify a particular comparison type. Once the collation service is called, it will return a compare result and a return and reason code. For two given Unicode input strings A and B, the compare result shows how one string is related to the other in the following way:

- -1, if A < B
- 0, if A = B
- 1, if A > B

The compare result and return codes are returned in the fields CUNBOPRM_Result, CUNBOPRM_Return_Code, and CUNBOPRM_Reason_code (for 31-bit), or CUN4BOPR_Result, CUN4BOPR_Return_Code and CUN4BOPR_Reason_code (for 64-bit),

respectively. To set alternate weighting options and a collation level, parameter fields CUNBOPRM_Mask and CUNBOPRM_Coll_Level (for 31-bit) or CUN4BOPR_Mask and CUN4BOPR_Coll_Level (for 64-bit) are used, respectively.

For more information on how to use these fields, see “Description of parameters in area CUNBOPRM” on page 113.

The two input Unicode strings to be compared are set in the same way as the other Unicode Services source buffers. A buffer pointer, length, and ALET are set for each source buffer.

The target buffers that are used to hold the converted bytes in the other Unicode services are not needed to be set in this case. That is because no bytes will be converted, except if the CUNBOPRM_Norm_Type or CUN4BOPR_Norm_Type field is equal to NFD, NFKD, NFC or NFKC.

For UCA400R1 and UCA410 versions, only NFD are supported. If Collation API is set with version 2 and there is an NF (Normalization Form) set differently from NFD, the NF will be ignored and Normalization will no longer be considered. Also RC = CUN_RC_WARN, RS = CUN_RS_INVALID_NORMALIZATION_VALUE will be set, even the process continues without any Normalization Form.

The results obtained from the comparison are returned in the result, return and reason code fields as described in the paragraph above. The work buffers are used as auxiliary buffers to hold data during the collation process. The work buffers should always be set in each collation call with the sufficient length needed during the collation process, otherwise a work buffer error will result.

For more information about the target and work buffers, see “Target buffer length considerations” on page 151 and “Work buffer length considerations” on page 150.

2. **Sort Key**

A sort key, or sort key vector, is a collection of weights for a given Unicode string which can be binary compared against another sort key to produce a compare result.

Sort keys can result from the collation process if the user sets the parameter area field CUNBOPRM_Coll_Mask or CUN4BOPR_Coll_Mask with constant CUNBOPRM_MASK_SK (see call samples). An associated comparison level and alternate weighting option can be specified by the user to form a particular sort key. Also, as part of new settings for Collation versions UCA400R1 and UCA410 consider the long and short path for sort key generation settings.

The sort key can be considered a “compare file”, because it can be created as a data set if properly specified by the user. The usefulness of a sort key is that once created for an input string, it can be kept and used repeatedly by the caller in binary comparisons with other sort keys. This can represent a performance advantage for the caller, because in this case there would be no need to call the collation services, but only perform a binary comparison with the caller’s preferred compare routine.

A sort key for a given Unicode character is formed by reading and processing the level weights found in the AllKeys.txt file provided by the Unicode consortium at: <http://www.Unicode.org/Unicode/reports/tr10/allkeys.txt>. Collation version 3.0.1 follows sort key generation as described on the Unicode Consortium TR#10, while recent Collation versions UCA400R1 and UCA410 do not due to tailoring features.

In order to use this collation functionality, the target buffers must be set by the caller in addition to the source and work buffers. The target buffers will hold the resulting sort key for their respective source buffers. Both or only one sort key

can be generated on each call to the collation services. To assume that one of the source buffers is not being used you must set its length at zero.

If you plan on using your own binary compare algorithms for sort keys, it is important you can interpret the sort key format. This is explained in “Sort key vector format” on page 149. The size of the sort key is determined by the collation level chosen. The greater the collation level, the longer the sort key will be.

z/OS Unicode Services collation does not provide a way of making a binary comparison for any pair of sort keys provided by the user. It is the user's responsibility to do the binary comparisons. If, after a call to z/OS, collation returns a zero return code, you can check for the sort key left in the target buffer(s). Otherwise, you must interpret the return and reason code, and retry a collation call after taking the appropriate steps.

For new Collation versions UCA400R1 and UCA410, sort key weights have different values than their respective versions from the DUCET (Default Unicode Collation Element Table - <http://www.unicode.org/Public/UCA/latest/allkeys.txt>) because they were modified for tailoring reasons (Locales or User Collation Rules - UCR).

According to each UCA (Unicode Collation Algorithm) version and settings (Locales or UCR) the Sort keys might contain different weights and then comparisons between different UCA version sort keys, in combination with some Locales or UCR, might return with an undesired comparison result. A good practice to avoid undesired results with sort key previously generated would be making sort key comparisons if and only if they comes from the same settings, that is, same UCA version, Locale, Collation Level, case options, etc. Otherwise, results might be inconsistent.

General considerations

A successful call to collation always returns a valid collation handle. This handle can be used as a fast path when recalling the collation services, because it specifies a direct access to the collation tables. IBM recommends providing the collation handle if successive collation calls are to be performed. If the caller only desires to request a collation handle, the fields CUNBOPRM_Get_New_Handle or CUN4BOPR_Get_New_Handle must be set to X'80'. See description of the field CUNBOPRM_Flag1 in “Description of parameters in area CUNBOPRM” on page 113. A sample program, CUNSOSMC, is provided in SYS1.SAMPLIB.

The caller can put the source parameters in any data space. To allow the service to access data not in primary space, an ALET must be specified. An ALET of 0 indicates that the data is in the primary address space (default value), which is the case for most callers.

A dynamic data area (DDA) must always be specified. The required length is defined by constant CUNBOPRM_DDA_Req or CUN4BOPR_DDA_Req. Refer to the interface definition file (CUNBOIDF).

Restrictions for the calling environment

The following table lists the restrictions for calling the collation service.

Table 18. Restrictions for the calling environment

Property	Restriction
Authorization	Problem state or supervisor state, and any PSW key
Dispatchable unit mode	Task or SRB
Cross memory mode	Any PASN, HASN, or SASN
AMODE	31-bit or 64-bit
ASC mode	Called in primary mode but exploiting AR mode
Interrupt status	Enabled for I/O and external interrupts
Locks	May be held by the caller, but not required to hold any
Control parameters	Must be in the primary address space
Recovery environment	Provided exclusively by the caller of the conversion services

Using the C interface

This is the syntax call in C for calling the stub routine **CUNLOCOL** (collation). The mapping of the parameter area supplied by the header file `cunhc.h` is listed in “Mapping of parameters in C” on page 101.

```

/* Includes section                                     */
.....
#include <string.h>
#include <cunhc.h>
.....

/* Constants section                                     */

#define SLEN 10
#define WLEN 80
#define TLEN 80

/* Declaration section                                   */
/*      Group 1                                           */
unsigned char Sourcebuffer1 [SLEN ] = {
/* H E L L O */
/* ----- */
'\x00', '\x48', '\x00', '\x45', '\x00', '\x4C', '\x00', '\x4C', '\x00', '\x4F'
};
unsigned char Workbuffer1 [WLEN ];
unsigned char Targetbuffer1 [TLEN ];
/*      Group 2                                           */
unsigned char Sourcebuffer2 [SLEN ] = {
/* H E L L O */
/* ----- */
'\x00', '\x48', '\x00', '\x45', '\x00', '\x4C', '\x00', '\x6C', '\x00', '\x4F'
};

unsigned char Workbuffer2 [WLEN ];
unsigned char Targetbuffer2 [TLEN ];
/*      DDA                                           */
unsigned char DDA [CUNBOPRM_DDA_REQ ];

/* Declaring a user collation                             */

```

Collation

```

/* parameter area */
CUNBOPRM myparm = {CUNBOPRM_DEFAULT};
/* Making addressables PA buffers and */
/* setting buffers length */
myparm.Src1_Buf_Ptr=Sourcebuffer1;
myparm.Src1_Buf_Len=SLen;

myparm.Src2_Buf_Ptr=Sourcebuffer2;
myparm.Src2_Buf_Len=SLen;

myparm.Wrk1_Buf_Ptr=Workbuffer1;
myparm.Wrk2_Buf_Len=WLEN;

myparm.Wrk_Buf_Ptr=Workbuffer2;
myparm.Wrk_Buf_Len=WLEN;

myparm.Targ1_Buf_Ptr=Targetbuffer1;
myparm.Targ2_Buf_Len=TLEN;

myparm.Targ_Buf_Ptr=Targetbuffer2;
myparm.Targ_Buf_Len=TLEN;

myparm.DDA_Buf_Ptr=DDA;
myparm.DDA_Buf_Len=CUNBOPRM_DDA_REQ;

/* Set collation */
/* Level 1 = CUNBOPRM_PRIMARY */
myparm.Coll_Level = CUNBOPRM_PRIMARY;
/* Set collation scheme rules */
myparm.Coll_Mask[0] = CUNBOPRM_MASK_DEFAULT;
/* Calling collation service */
CUNLOCOL ( & myparm );
if(myparm.Return_Code == CUN_RC_OK) then
  If (myparm.Coll_Result = 0) then
    ..... /* SourceBuffer1 = SourceBuffer2 */
  else If (myparm.Coll_Result < 0) then
    ..... /* SourceBuffer1 < SourceBuffer2 */
  else
    ..... /* SourceBuffer1 > SourceBuffer2 */
else
  ..... /* an error had occurred */

```

The sample below shows how to use "long path" settings to call current Unicode Collation Version 4.0.1 (UCA401). For new collation features, the following interfaces can be used:

```

/* Includes section */
.....
#include <string.h>
#include <cunhc.h>
.....

/* Constants section */
#define SLEN 10
#define WLEN 80
#define TLEN 80

/* Declaration section */
/* Group 1 */
unsigned char Sourcebuffer1 [SLEN] = {
/* H E L L O */
/* ----- */
'\x00', '\x48', '\x00', '\x45', '\x00', '\x4C', '\x00', '\x4C', '\x00', '\x4F'
};
unsigned char Workbuffer1 [WLEN];
unsigned char Targetbuffer1 [TLEN];
/* Group 2 */

```

```

unsigned char Sourcebuffer2 [SLEN ] = {
/* H E L L O */
/* -----
'\x00','\x48','\x00','\x45','\x00','\x4C','\x00','\x6C','\x00','\x4F'
*/
};
unsigned char Workbuffer2 [WLEN ];
unsigned char Targetbuffer2 [TLEN ];
/* DDA */
unsigned char DDA [CUNBOPRM_DDA_REQ ];
/* Setting Collation PA version as 2 */
myparm.Version = CUNBOPRM_VERSION2;
/* Making addressables PA buffers and
/* setting buffers length */
myparm.Src1_Buf_Ptr=Sourcebuffer1;
myparm.Src1_Buf_Len=SLEN;
myparm.Src2_Buf_Ptr=Sourcebuffer2;
myparm.Src2_Buf_Len=SLEN;
myparm.Wrk1_Buf_Ptr=Workbuffer1;
myparm.Wrk2_Buf_Len=WLEN;
myparm.Wrk_Buf_Ptr=Workbuffer2;
myparm.Wrk_Buf_Len=WLEN;
myparm.Targ1_Buf_Ptr=Targetbuffer1;
myparm.Targ2_Buf_Len=TLEN;
myparm.Targ_Buf_Ptr=Targetbuffer2;
myparm.Targ_Buf_Len=TLEN;
myparm.DDA_Buf_Ptr=DDA;
myparm.DDA_Buf_Len=CUNBOPRM_DDA_REQ;

/* *****
/* Long path Collation settings */
/* *****

/* Collation PA version */
MyCollParm.Version = CUNBOPRM_VERSION2;
/* Coll mask settings */

MyCollParm.Coll_Mask.Variable_Opt = CUNBOPRM_MASK_NAVCE;
MyCollParm.Coll_Mask.Cmp_Order = CUNBOPRM_MASK_FORWARD;
MyCollParm.Coll_Mask.SKey_Opt = CUNBOPRM_MASK_nSK;
MyCollParm.Coll_Mask.Norm_Type = CUNBOPRM_MASK_nNORM;
MyCollParm.Coll_Mask.SKey_and_Cmp = CUNBOPRM_MASK_SKey_and_Cmp_OFF;

/* Coll Level settings */
MyCollParm.Coll_Level = CUNBOPRM_TERTIARY;
/* UCA version */
MyCollParm.UCA_Ver[0] = CUNBOPRM_UCA400R1;
/* Case Options settings */
MyCollParm.Case_Options.Case_First =
CUNBOPRM_CASE_OPTIONS_Case_First_Default;
MyCollParm.Case_Options.Case_Level=
CUNBOPRM_CASE_OPTIONS_Case_Level_OFF;
/* Hiragana settings */
MyCollParm.Special.Hiragana = CUNBOPRM_CASE_SPECIAL_Hiragana_OFF;

/* Locale Settings */
strcpy(MyCollParm.Locale.Language,"EN");
strcpy(MyCollParm.Locale.Region,"US");
strcpy(MyCollParm.Locale.Variant,"POSIX");

/* Calling collation service */
CUNLOCOL ( & myparm );
if(myparm.Return_Code == CUN_RC_OK) then
  If (myparm.Coll_Result = 0) then
    ..... /* SourceBuffer1 = SourceBuffer2 */
  else If (myparm.Coll_Result < 0) then
    ..... /* SourceBuffer1 < SourceBuffer2 */

```

Collation

```

        else
            ..... /* SourceBuffer1 > SourceBuffer2 */
    else
        ..... /* an error had occurred */

```

Calling Collation Service UCA400R1 short path settings:

```

/* Includes section */
.....
#include <string.h>
#include <cunhc.h>
.....

/* Constants section */

#define SLEN 10
#define WLEN 80
#define TLEN 80

/* Declaration section */
/* Group 1 */
unsigned char Sourcebuffer1 [SLEN] = {
/* H E L L O */
/* ----- */
'\x00','\x48','\x00','\x45','\x00','\x4C','\x00','\x4C','\x00','\x4F'
};
unsigned char Workbuffer1 [WLEN];
unsigned char Targetbuffer1 [TLEN];
/* Group 2 */
unsigned char Sourcebuffer2 [SLEN] = {
/* H E L L O */
/* ----- */
'\x00','\x48','\x00','\x45','\x00','\x4C','\x00','\x6C','\x00','\x4F'
};
unsigned char Workbuffer2 [WLEN];
unsigned char Targetbuffer2 [TLEN];
/* DDA */
unsigned char DDA [CUNBOPRM_DDA_REQ];
/* Setting Collation PA version as 2 */
myparm.Version = CUNBOPRM_VERSION2;
/* Making addressables PA buffers and */
/* setting buffers length */
myparm.Src1_Buf_Ptr=Sourcebuffer1;
myparm.Src1_Buf_Len=SLEN;
myparm.Src2_Buf_Ptr=Sourcebuffer2;
myparm.Src2_Buf_Len=SLEN;
myparm.Wrk1_Buf_Ptr=Workbuffer1;
myparm.Wrk2_Buf_Len=WLEN;
myparm.Wrk_Buf_Ptr=Workbuffer2;
myparm.Wrk_Buf_Len=WLEN;
myparm.DDA_Buf_Ptr=DDA;
myparm.DDA_Buf_Len=CUNBOPRM_DDA_REQ;
/* Setting Collation Keywords as */
/* short path settings */
strcpy(myparm.Collation_Keyword,"UCA400R1_LEN_RUS_VPOSIX_S3");

/***** Collation Keywords Reference *****/
/* Sample: */
/*
/* UCA400R1_LEN_RGB_PREEURO_S1_KX_CD_AD_T0301xxxx_ND_FD_HD
/* ++ ++ ++++++ + + + + ++++++ +++++ + + +
/* ?? ?? ?????? 1 X X N ?????? ???? D D D
/* 2 O L S X X X
/* 3 D U D 0 0 0
/* 4 D
/* I
/* D
/*

```

```

/* Collation Keywords Reference */
/* +-----+-----+-----+ */
/* |Attribute Name|Key |Possible Values | */
/* +-----+-----+-----+ */
/* |Locale        |L.R.V |<Locale>      | */
/* +-----+-----+-----+ */
/* |Strength      |S      | 1, 2, 3, 4, I, D | */
/* +-----+-----+-----+ */
/* |Case_Level    |K      | X, 0, D          | */
/* +-----+-----+-----+ */
/* |Case_First    |C      | X, L, U, D       | */
/* +-----+-----+-----+ */
/* |Alternate     |A      | N, S, D          | */
/* +-----+-----+-----+ */
/* |Variable_Top  |T      |                  | */
/* +-----+-----+-----+ */
/* |Normalization|N      |X, 0, D          | */
/* +-----+-----+-----+ */
/* |French        |F      |X, 0, D          | */
/* +-----+-----+-----+ */
/* |Hiragana      |H      |X, 0, D          | */
/* +-----+-----+-----+ */
/* */
/* Collation Keyword values description */
/* +-----+-----+ */
/* |Description  |Abbreviation| */
/* +-----+-----+ */
/* |Default      |D          | */
/* |On           |0          | */
/* |Off          |X          | */
/* |Primary      |1          | */
/* |Secondary    |2          | */
/* |Tertiary     |3          | */
/* |Quaternary   |4          | */
/* |Identical    |I          | */
/* |Shifted      |S          | */
/* |Non-Ignorable|N          | */
/* |Lower-First  |L          | */
/* |Upper-First  |U          | */
/* +-----+-----+ */
/* */
/*****
/* Calling collation service */
CUNLOCOL ( & myparm );
if(myparm.Return_Code == CUN_RC_OK) then
  If (myparm.Coll_Result = 0) then
    ..... /* SourceBuffer1 = SourceBuffer2 */
  else If (myparm.Coll_Result < 0) then
    ..... /* SourceBuffer1 < SourceBuffer2 */
  else
    ..... /* SourceBuffer1 > SourceBuffer2 */
else
  ..... /* an error had occurred */

```

Mapping of parameters in C

A C header file is supplied (cunhc.h) that contains the function prototypes, default values, and constants to call the collation service. The structure tagCUNBOPRM contains the collation user parameter area mapped in C.

31-bit mapping

```

typedef struct tagCUNBOPRM {
    long        Version;           /* Structure version number */
    long        Length;           /* Length of structure */
    long        Res1;             /* Reserved */
    void *      Src1_Buf_Ptr;      /* Pointer to Source 1 */
    unsigned long Src1_Buf_ALET;   /* ALET of source buffer 1 */
    unsigned long Src1_Buf_Len;   /* Length of source data 1 */
    long        Res2;             /* Reserved */
    void *      Src2_Buf_Ptr;      /* Pointer to Source2 */
    unsigned long Src2_Buf_ALET;   /* ALET of source buffer 2 */
    unsigned long Src2_Buf_Len;   /* Length of source data 2 */
    long        Res3;             /* Reserved */
    void *      Targ1_Buf_Ptr;     /* Pointer to Target 1 */
    unsigned long Targ1_Buf_ALET;  /* ALET of target buffer 1 */
    unsigned long Targ1_Buf_Len;  /* Length of target data 1 */
    long        Res4;             /* Reserved */
    void *      Targ2_Buf_Ptr;     /* Pointer to target 2 */
    unsigned long Targ2_Buf_ALET;  /* ALET of target buffer 2 */
    unsigned long Targ2_Buf_Len;  /* Length of target data 2 */
    char        Coll_Handle[64];   /* Collation handle */
    unsigned char Coll_Level;      /* Collation Level type */
    unsigned char Res5[7];        /* Reserved */
    long        Res6;             /* Reserved */
    void *      Wrk1_Buf_Ptr;      /* Pointer to work1 buffer */
    unsigned long Wrk1_Buf_ALET;   /* ALET of work1 buffer */
    unsigned long Wrk1_Buf_Len;   /* Length of work1 buffer */
    long        Res7;             /* Reserved */
    void *      Wrk2_Buf_Ptr;      /* Pointer to work2 buffer */
    unsigned long Wrk2_Buf_ALET;   /* ALET of work2 buffer */
    unsigned long Wrk2_Buf_Len;   /* Length of work2 buffer */
    long        Res8;             /* Reserved */
    void *      DDA_Buf_Ptr;       /* Pointer to dynamic data area */
    unsigned long DDA_Buf_ALET;    /* ALET of DDA */
    unsigned long DDA_Buf_Len;    /* Length of DDA */
    struct {
        int        Inv_Handle      : 1, /* Invalid handle action: */
                                           /* 0 = Terminate with error */
                                           /* 1 = Get new handle and */
        int        Get_New_Handle  : 1, /* Get a new handle */
                                           /* Source Character */
                                           /* 0 = Get/Use a handle and */
                                           /* continue with the service */
                                           /* 1 = Get handle and return */
                                           /* to the caller */
        int        Page_Fix       : 1, /* Page Fixing: */
                                           /* 0=System storage */
                                           /* 1=Page Fixing. */
        int        : 5;
    } Flag1;
    unsigned char Res9[1];        /* Reserved */
    struct {
        int        Variable_Opt   : 3, /* Where : */
                                           /* 0 - Shifted */
                                           /* 1 - Blanked */
                                           /* 10 - Non Blanked */
                                           /* 11 - Shift-Trimmed and */
        int        Cmp_Order      : 1, /* Where : */
                                           /* 0 - Forward */
                                           /* 1 - Backward (French) */
        int        SKey_Opt       : 1, /* Where: */
                                           /* 0 - Not Get Sort Key */
                                           /* 1 - Get Sort Key */
        int        Norm_Type      : 3, /* Normalization Form */
                                           /* 000 - No Apply Normalization */
                                           /* 001 - Apply NFD */
    }

```



```

/* 010 - Apply NFC */
/* 011 - Apply NFKD */
/* 100 - Apply NFKC */
SKey_and_Cmp : 1, /* Make binary comparison */
/* (CUNBOPRM_RESULT), if and */
/* only if, CUN4BOPR_SKey_Opt */
/* is ON: */
/* 0 - Do not perform binary */
/* comparison */
/* 1 - Perform binary comparison */
: 7; /* Padding */
} Coll_Mask;
signed long Coll_Result; /* Collation Mask */
long Return_Code; /* Collation Result */
long Reason_Code; /* Return code */
unsigned char UCA_Ver[1]; /* Reason code */
unsigned char Res10[2]; /* UCA Version */
struct {
    int Case_First : 8, /* Where: */
/* 000 - Default */
/* 001 - Upper First */
/* 010 - Lower First */
Case_Level : 1, /* Where: */
/* 0 - Default */
/* 1 - Primary Level will */
/* ignore accent but not */
/* case */
: 7; /* Padding */
} Case_Options;
struct {
    int Hiragana : 1, /* Distinguish between Japanese */
/* hiragana and Katakana chars */
/* Where: */
/* 0 - Default */
/* 1 - Conform to the */
/* Japanese JIS X 4061 */
/* standard with Primary */
/* Level */
: 7; /* Reserved */
} Special;
unsigned char Res11[2]; /* Padding */
unsigned long Var_Top; /* Variable Top - UTF16BE */
struct {
    char Language [ 2]; /* Language */
    char Underscore1 [ 1]; /* Underscore */
    char Region [ 2]; /* Region */
    char Underscore2 [ 1]; /* Underscore */
    char Variant [26]; /* Variant */
} Locale;
unsigned char Res12[2]; /* Padding */
unsigned char Collation_Keyword[64]; /* Collation Keyword - ICU */
/* set short form */
unsigned char DSName[44]; /* Data Set Name */
unsigned char Res13[4]; /* Padding */
unsigned char Collation_Rules_File[8]; /* Member */
unsigned char Collation_Rules_Vol[6]; /* Data Set Name Volume */
unsigned char Res14[2]; /* Padding */
} CUNBOPRM;

```

64-bit mapping

```

typedef struct tagCUN4BOPR {
    unsigned int Version; /* Structure version number */
    unsigned int Length; /* Length of structure */
    void * Src1_Buf_Ptr; /* Pointer to Source 1 */
    unsigned int Res1; /* Reserved */
    unsigned int Src1_Buf_ALET; /* ALET of source buffer 1 */
}

```

Collation

```

unsigned long Src1_Buf_Len;      /* Length of source data 1 */
void * Src2_Buf_Ptr;           /* Pointer to Source2 */
unsigned int Res2;              /* Reserved */
unsigned int Src2_Buf_ALET;     /* ALET of source buffer 2 */
unsigned long Src2_Buf_Len;     /* Length of source data 2 */
void * Targ1_Buf_Ptr;           /* Pointer to Target 1 */
unsigned int Res3;              /* Reserved */
unsigned int Targ1_Buf_ALET;     /* ALET of target buffer 1 */
unsigned long Targ1_Buf_Len;     /* Length of target data 1 */
void * Targ2_Buf_Ptr;           /* Pointer to target 2 */
unsigned int Res4;              /* Reserved */
unsigned int Targ2_Buf_ALET;     /* ALET of target buffer 2 */
unsigned long Targ2_Buf_Len;     /* Length of target data 2 */
char Coll_Handle[64];           /* Collation handle */
unsigned char Coll_Level;        /* Collation Level type */
unsigned char Res5[7];           /* Reserved */
void * Wrk1_Buf_Ptr;            /* Pointer to work1 buffer */
unsigned int Res6;              /* Reserved */
unsigned int Wrk1_Buf_ALET;      /* ALET of work1 buffer */
unsigned long Wrk1_Buf_Len;      /* Length of work1 buffer */
void * Wrk2_Buf_Ptr;            /* Pointer to work2 buffer */
unsigned int Res7;              /* Reserved */
unsigned int Wrk2_Buf_ALET;      /* ALET of work2 buffer */
unsigned long Wrk2_Buf_Len;      /* Length of work2 buffer */
void * DDA_Buf_Ptr;             /* Pointer to dynamic data area */
unsigned int DDA_Buf_ALET;       /* ALET of DDA */
unsigned int DDA_Buf_Len;        /* Length of DDA */
struct {
    int Inv_Handle : 1, /* Invalid handle action: */
        /* 0 = Terminate with error */
        /* 1 = Get new handle and */
        Get_New_Handle : 1, /* Get a new handle */
        /* Source Character */
        /* 0 = Get/Use a handle and */
        /* continue with the service */
        /* 1 = Get handle and return */
        /* to the caller */
        Page_Fix : 1, /* Page Fixing: */
        /* 0=System storage */
        /* 1=Page Fixing. */
        : 5; /* FLAG Byte 1 set by caller */
    unsigned char Res8; /* Reserved */
    struct {
        int Variable_Opt : 3, /* Where : */
            /* 0 - Shifted */
            /* 1 - Blanked */
            /* 10 - Non Blanked */
            /* 11 - Shift-Trimmed and */
            Cmp_Order : 1, /* Where : */
            /* 0 - Forward */
            /* 1 - Backward (French) */
            SKey_Opt : 1, /* Where: */
            /* 0 - Not Get Sort Key */
            /* 1 - Get Sort Key */
            Norm_Type : 3, /* Normalization Form */
            /* 000 - No Apply Normalization */
            /* 001 - Apply NFD */
            /* 010 - Apply NFC */
            /* 011 - Apply NFKD */
            /* 100 - Apply NFKC */
            SKey_and_Cmp : 1, /* Make binary comparison */
            /* (CUNBOPRM_RESULT), if and */
            /* only if, CUN4BOPR_SKey_Opt */
            /* is ON: */
            /* 0- Do not perform binary */
            /* comparison */

```

```

/* 1- Perform binary comparison*/
: 7; /* Padding */
} Coll_Mask; /* Collation Mask */
int Coll_Result; /* Collation Result */
unsigned int Return_Code; /* Return code */
unsigned int Reason_Code; /* Reason code */
unsigned char UCA_Ver[1]; /* UCA Version */
unsigned char Res9[2]; /* Padding */
struct {

    int Case_First : 8, /* Where: */
        /* 000 - Default */
        /* 001 - Upper First */
        /* 010 - Lower First */
        Case_Level : 1, /* Where: */
        /* 0 - Default */
        /* 1 - Primary Level will */
        /* ignore accent but not */
        /* case */
        : 7; /* Padding */
} Case_Options;
struct {
    int Hiragana : 1, /* Distinguish between Japanese */
        /* hiragana and Katakana chars */
        /* Where: */
        /* 0 - Default */
        /* 1 - Conform to the */
        /* Japanese JIS X 4061 */
        /* standard with Primary */
        /* Level */
        : 7; /* Reserved */
} Special;
unsigned char Res10[2]; /* Padding */
unsigned long Var_Top; /* Variable Top - UTF16BE */
struct {
    char Language [ 2]; /* Language */
    char Underscore1 [ 1]; /* Underscore */
    char Region [ 2]; /* Region */
    char Underscore2 [ 1]; /* Underscore */
    char Variant [26]; /* Variant */
} Locale;
unsigned char Res11[2]; /* Padding */
unsigned char Collation_Keyword[64]; /* Collation Keyword - ICU */
/* set short form */
unsigned char DSName[44]; /* Data Set Name */
unsigned char Res12[4]; /* Padding */
unsigned char Collation_Rules_File[8]; /* Member */
unsigned char Collation_Rules_Vol[6]; /* Data Set Name Volume */
unsigned char Res13[2]; /* Padding */
} CUN4BOPR;

```

Mapping of constants in C

Also, cunhc contains a group of constants to establish the Collation rules. These are the constants.

Group 1 - Collation level

These constants set up the Coll_Level, and must be specified individually.

DDA size

Collation

```
#ifdef _LP64
#define CUNBOPRM_DDA_REQ      8192
#else
#define CUNBOPRM_DDA_REQ      4096
#endif
```

Collation Parameter Area versions

```
#define CUNBOPRM_VERSION      1
#define CUNBOPRM_VERSION2    2
```

ALET Constant

```
#define CUNBOPRM_ALET        0
```

Collation Levels (also named Collation strengths)

```
#define CUNBOPRM_IDENTICAL    5
#define CUNBOPRM_PRIMARY      1
#define CUNBOPRM_SECONDARY    2
#define CUNBOPRM_TERTIARY     3
#define CUNBOPRM_QUATERNARY   4
#define CUNBOPRM_QUINARY      5
```

Collation Mask

```
#define CUNBOPRM_MASK_DEFAULT '\xE0' /* naVCE+Forward+nSK+nNorm */
```

Used for Variable_Opt field

```
#define CUNBOPRM_MASK_SHIFTED      0
#define CUNBOPRM_MASK_BLANKED      1
#define CUNBOPRM_MASK_nIGNORABLE    2
#define CUNBOPRM_MASK_STRIMMED      3
#define CUNBOPRM_MASK_NAVCE         14
```

Used for Cmp_Order field

```
#define CUNBOPRM_MASK_FORWARD      0
#define CUNBOPRM_MASK_BACKWARD     1
```

Used for SKey_Opt field

```
#define CUNBOPRM_MASK_nSK          0
#define CUNBOPRM_MASK_SK           1
```

Used for Norm_Type field

```
#define CUNBOPRM_MASK_nNORM        0
#define CUNBOPRM_MASK_NFD          1
#define CUNBOPRM_MASK_NFC          2
#define CUNBOPRM_MASK_NFKD         3
#define CUNBOPRM_MASK_NFKC         4
```

Used for SKey_and_Cmp field

```
#define CUNBOPRM_MASK_SKey_and_Cmp_OFF 0
#define CUNBOPRM_MASK_SKey_and_Cmp_ON  1
```

Used for Case_First field

```
#define CUNBOPRM_CASE_OPTIONS_Case_First_Default 0
#define CUNBOPRM_CASE_OPTIONS_Case_First_UPPER  1
#define CUNBOPRM_CASE_OPTIONS_Case_First_lower   2
```

Used for Case_Level field

```
#define CUNBOPRM_CASE_OPTIONS_Case_Level_OFF 0
#define CUNBOPRM_CASE_OPTIONS_Case_Level_ON  1
```

Used for Hiragana field

```
#define CUNBOPRM_CASE_SPECIAL_Hiragana_OFF      0
#define CUNBOPRM_CASE_SPECIAL_Hiragana_ON       1
```

Used for Handle bit fields

```
#define CUNBOPRM_FLAG1_DEFAULT      '\x00'
#define CUNBOPRM_FLAG1_Ret_If_Inv_Handle_ON '\x80'
#define CUNBOPRM_FLAG1_Get_New_Handle_ON '\x40'
```

Null Handle

```
#define CUNBOPRM_EMPTY_COLLHDL '\0','\0','\0','\0','\0','\0','\0','\0',\
                                '\0','\0','\0','\0','\0','\0','\0','\0',\
                                '\0','\0','\0','\0','\0','\0','\0','\0',\
                                '\0','\0','\0','\0','\0','\0','\0','\0',\
                                '\0','\0','\0','\0','\0','\0','\0','\0',\
                                '\0','\0','\0','\0','\0','\0','\0','\0',\
                                '\0','\0','\0','\0','\0','\0','\0','\0',\
                                '\0','\0','\0','\0','\0','\0','\0','\0'
```

UCA (Unicode Collation Algorithm) versions

```
#define CUNBOPRM_UCAempty      '\x00'
#define CUNBOPRM_UCA301      '\x01'
#define CUNBOPRM_UCA400R1    '\x02'
#define CUNBOPRM_UCA410      '\x03'
```

There is also a C example in the **CUNSOSMC** member in SYS1.SAMPLIB. For further sample information, see “Sample programs” on page 152.

Using the HLASM interface

This is the call syntax in HLASM for calling the stub routine **CUNLOCOL** for AMODE (31) and **CUN4LCOL** for AMODE (64). Sample programs, **CUNSOSMA** for 31-bit and **CUN4SOSA** for 64-bit, are provided in SYS1.SAMPLIB.

Following is an example of how you can invoke the collation service with the HLASM interface. You can find this sample in the samples library (SYS1.SAMPLIB) as **CUNSOSMA** for 31-bit and **CUN4SOSA** for 64-bit.

For AMODE (31)

```
-----1-----2-----3-----4-----5-----6-----7--
.....
*****
* PREPARE PARAMETER AREA FOR CALL TO THE CONVERSION ROUTINES *
*****
      SPACE 1
      GETMAIN ..... Obtain storage for parameter area
*      in primary address space.
      LR R4,R1                      ! Save parameter area
*                                  ! address
      LA  R8,PARMAREA                ! GET PARAMETER AREA ADDR
      USING CUNBOPRM,R8              ! ESTABLISH ADDRESSABILITY
      SPACE 1
*                                  ! CLEAR PARAMETER AREA
      XC  CUNBOPRM(CUNBOPRM_LEN),CUNBOPRM
      LA  R0,CUNBOPRM_VER             ! GET ACTUAL VERSION
      ST  R0,CUNBOPRM_VERSION         ! STORE INTO PARAMETER
      LA  R0,CUNBOPRM_LEN             ! GET ACTUAL LENGTH
      ST  R0,CUNBOPRM_LENGTH          ! STORE INTO PARAMETER
*                                  /*****
*                                  /* Setting source buffers */
```

Collation

```

*                                     /******/
SPACE 2
MVC  CUNBOPRM_SRC1_BUF_PTR,ASRC1    ! SOURCE1 BUFFER PTR
MVC  CUNBOPRM_SRC1_BUF_ALET,=F'0'   ! PRIMARY MODE ALET
MVC  CUNBOPRM_SRC1_BUF_LEN,SRC1LEN  ! SOURCE1 BUFFER LENGTH
SPACE 1
MVC  CUNBOPRM_SRC2_BUF_PTR,ASRC2    ! SOURCE2 BUFFER PTR
MVC  CUNBOPRM_SRC2_BUF_ALET,=F'0'   ! PRIMARY MODE ALET
MVC  CUNBOPRM_SRC2_BUF_LEN,SRC2LEN  ! SOURCE2 BUFFER LENGTH
*                                     /******/
*                                     /*      Setting Work buffers      */
*                                     /******/
.....
*                                     /******/
*                                     /*      Setting Target buffers      */
*                                     /******/
.....
*****
*                                     IMPORTANT: DDA IS ALWAYS REQUIRED      *
*****
*                                     /******/
*                                     /*      Setting DDA buffers      */
*                                     /******/
SPACE 1
MVC  CUN4BOPR_DDA_BUF_PTR,ADDA      ! DYNAMIC DATA AREA
MVC  CUN4BOPR_DDA_BUF_ALET,=F'0'    ! PRIMARY MODE ALET
LG  R0,=A(CUN4BOPR_DDA_REQ)         ! GET DDA LENGTH
STG  R0,CUN4BOPR_DDA_BUF_LEN        ! STORE INTO PARAMETER
SPACE 1
*****
*                                     NOW FILL PARAMETER AREA      *
*****
SPACE 1
LA  R0,CUNBOPRM_TERTIARY            ! GET COLLATION LEVEL
STC  R0,CUNBOPRM_COLL_LEVEL         ! STORE TO PARAMETER AREA
SPACE 1
LA  R0,Mask_Default                ! SET COLLATION MASK
STC  R0,CUNBOPRM_MASK              ! STORE TO PARAMETER AREA
*                                     /*      Copying to source 1      */
SPACE 1
L  R2,CUNBOPRM_SRC1_BUF_PTR         ! GET SRC1 BUFFER ADDRESS
L  R3,STR1LEN                       ! GET ACTUAL TO-LENGTH
L  R4,ASTRING1                      ! GET DATA BUFFER ADDRESS
LR  R5,R3                           ! GET ACTUAL FROM-LENGTH
ST  R5,CUNBOPRM_SRC1_BUF_LEN        ! UPDATE SRC BUFFER LENGTH
MVCL R2,R4                          ! MOVE DATA TO SOURCE BUFF
*                                     /*      Copying to source 2      */
SPACE 1
L  R2,CUNBOPRM_SRC2_BUF_PTR         ! GET SRC1 BUFFER ADDRESS
L  R3,STR2LEN                       ! GET ACTUAL TO-LENGTH
L  R4,ASTRING2                      ! GET DATA BUFFER ADDRESS
LR  R5,R3                           ! GET ACTUAL FROM-LENGTH
ST  R5,CUNBOPRM_SRC2_BUF_LEN        ! UPDATE SRC BUFFER LENGTH
MVCL R2,R4                          ! MOVE DATA TO SOURCE BUFF
*****
*                                     CALLING THE COLLATION SERVICE      *
*****
SPACE 1
CALL CUNLOCOL,PARMAREA
SPACE 1
*****
*      Check CUNBOPRM_Return_Code and CUNBOPRM_Reason_Code      *
*      and CUNBOPRM_Result; where                                *
*      if CUNBOPRM_Result = -1, then String1 < String2;          *
*      if CUNBOPRM_Result = 0, then String1 = String2;           *

```

```

*          if CUNBOPRM_Result = 1, then String1 > String2;          *
*                                                                    *
*****
.....

For AMODE (64)

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
.....
*****
* PREPARE PARAMETER AREA FOR CALL TO THE CONVERSION ROUTINES      *
*****
      SPACE 1
      GETMAIN ..... Obtain storage for parameter area
*      in primary address space.
      LR R4,R1                                ! Save parameter area
*                                          ! address
      LA  R8,PARMAREA                          ! GET PARAMETER AREA ADDR
      USING CUN4BOPR,R8                        ! ESTABLISH ADDRESSABILITY
      SPACE 1
*                                          ! CLEAR PARAMETER AREA
      XC  CUN4BOPR(CUN4BOPR_LEN),CUN4BOPR
      LA  R0,CUN4BOPR_VER                      ! GET ACTUAL VERSION
      ST  R0,CUN4BOPR_VERSION                  ! STORE INTO PARAMETER
      LA  R0,CUN4BOPR_LEN                      ! GET ACTUAL LENGTH
      ST  R0,CUN4BOPR_LENGTH                  ! STORE INTO PARAMETER
*                                          /*****
*                                          /*      Setting source buffers      */
*                                          /*****

      SPACE 2
      MVC CUN4BOPR_SRC1_BUF_PTR,ASRC1          ! SOURCE1 BUFFER PTR
      MVC CUN4BOPR_SRC1_BUF_ALET,=F'0'         ! PRIMARY MODE ALET
      MVC CUN4BOPR_SRC1_BUF_LEN,SRC1LEN        ! SOURCE1 BUFFER LENGTH
      SPACE 1
      MVC CUN4BOPR_SRC2_BUF_PTR,ASRC2          ! SOURCE2 BUFFER PTR
      MVC CUN4BOPR_SRC2_BUF_ALET,=F'0'         ! PRIMARY MODE ALET
      MVC CUN4BOPR_SRC2_BUF_LEN,SRC2LEN        ! SOURCE2 BUFFER LENGTH
*                                          /*****
*                                          /*      Setting Work buffers      */
*                                          /*****

.....
*                                          /*****
*                                          /*      Setting Target buffers      */
*                                          /*****

.....
*****
*          IMPORTANT: DDA IS ALWAYS REQUIRED                          *
*****
*                                          /*****
*                                          /*      Setting DDA buffers      */
*                                          /*****

      SPACE 1
      MVC CUN4BOPR_DDA_BUF_PTR,ADDA            ! DYNAMIC DATA AREA
      MVC CUN4BOPR_DDA_BUF_ALET,=F'0'          ! PRIMARY MODE ALET
      L   R0,=A(CUN4BOPR_DDA_REQ)              ! GET DDA LENGTH
      ST  R0,CUN4BOPR_DDA_BUF_LEN              ! STORE INTO PARAMETER
      SPACE 1
*****
*          NOW FILL PARAMETER AREA                                  *
*****
      SPACE 1
      LA  R0,CUN4BOPR_TERTIARY                  ! GET COLLATION LEVEL
      STC R0,CUN4BOPR_COLL_LEVEL                ! STORE TO PARAMETER AREA
      SPACE 1
      LA  R0,Mask_Default                      ! SET COLLATION MASK
      STC R0,CUN4BOPR_MASK                    ! STORE TO PARAMETER AREA
*                                          /*      Copying to source 1      */

```

Collation

```

SPACE 1
L    R2,CUN4BOPR_SRC1_BUF_PTR    ! GET SRC1 BUFFER ADDRESS
L    R3,STR1LEN                  ! GET ACTUAL TO-LENGTH
L    R4,ASTRING1                 ! GET DATA BUFFER ADDRESS
LR   R5,R3                      ! GET ACTUAL FROM-LENGTH
ST   R5,CUN4BOPR_SRC1_BUF_LEN    ! UPDATE SRC BUFFER LENGTH
MVCL R2,R4                      ! MOVE DATA TO SOURCE BUFF
*                                /*      Copying to source 2      */
SPACE 1
L    R2,CUN4BOPR_SRC2_BUF_PTR    ! GET SRC1 BUFFER ADDRESS
L    R3,STR2LEN                  ! GET ACTUAL TO-LENGTH
L    R4,ASTRING2                 ! GET DATA BUFFER ADDRESS
LR   R5,R3                      ! GET ACTUAL FROM-LENGTH
ST   R5,CUN4BOPR_SRC2_BUF_LEN    ! UPDATE SRC BUFFER LENGTH
MVCL R2,R4                      ! MOVE DATA TO SOURCE BUFF
*****
*                                CALLING THE COLLATION SERVICE      *
*****
SPACE 1
CALL CUN4LCOL,PARMAREA
SPACE 1
*****
*      Check CUN4BOPR_Return_Code and CUN4BOPR_Reason_Code      *
*      and CUN4BOPR_Result; where                                *
*      if CUN4BOPR_Result = -1, then String1 < String2;          *
*      if CUN4BOPR_Result = 0, then String1 = String2;          *
*      if CUN4BOPR_Result = 1, then String1 > String2;          *
*                                                                *
*****
.....

```

For more HLASM samples, see “Sample programs” on page 152.

Mapping of parameters for AMODE (31)

The mapping of the parameter areas is supplied by the interface definition file CUNBOIDF. This file is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

Table 19. Mapping of parameters in HLASM for collation AMODE (31)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	380	DWORD	CUNBOPRM	Parameter Area
0	(0)	UNSIGNED	4		CUNBOPRM_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUNBOPRM_Length	Parameter area Length
8	(8)	CHARACTER	4		*	Reserved for 64 bit
12	(C)	ADDRESS	4		CUNBOPRM_Src1_Buf_Ptr	Source1 buffer pointer
16	(10)	UNSIGNED	4		CUNBOPRM_Src1_Buf_ALET	Source1 buffer ALET
20	(14)	UNSIGNED	4		CUNBOPRM_Src1_Buf_Len	Source1 buffer length
24	(18)	CHARACTER	4		*	Reserved for 64 bit
28	(1C)	ADDRESS	4		CUNBOPRM_Src2_Buf_Ptr	Source2 buffer pointer
32	(20)	UNSIGNED	4		CUNBOPRM_Src2_Buf_ALET	Source2 buffer ALET
36	(24)	UNSIGNED	4		CUNBOPRM_Src2_Buf_Len	Source2 buffer length
40	(28)	CHARACTER	4		*	Reserved for 64 bit
44	(2C)	ADDRESS	4		CUNBOPRM_Targ1_Buf_Ptr	Target1 buffer pointer

Table 19. Mapping of parameters in HLASM for collation AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
48	(30)	UNSIGNED	4		CUNBOPRM_Targ1_Buf_ALET	Target1 buffer ALET
52	(34)	UNSIGNED	4		CUNBOPRM_Targ1_Buf_Len	Target1 buffer length
56	(38)	CHARACTER	4		*	Reserved for 64 bit
60	(3C)	ADDRESS	4		CUNBOPRM_Targ2_Buf_Ptr	Target2 buffer pointer
64	(40)	UNSIGNED	4		CUNBOPRM_Targ2_Buf_ALET	Target2 buffer ALET
68	(44)	UNSIGNED	4		CUNBOPRM_Targ2_Buf_Len	Target2 buffer length
72	(48)	CHARACTER	64	DWORD	CUNBOPRM_Coll_Handle	Collation handle
136	(88)	CHARACTER	1		CUNBOPRM_Coll_Level	Collation level
137	(89)	CHARACTER	7		*	Reserved
144	(90)	CHARACTER	4		*	Reserved for 64 bit
148	(94)	ADDRESS	4		CUNBOPRM_Wrk1_Buf_Ptr	Work1 buffer pointer
152	(98)	UNSIGNED	4		CUNBOPRM_Wrk1_Buf_ALET	Work1 buffer ALET
156	(9C)	UNSIGNED	4		CUNBOPRM_Wrk1_Buf_Len	Work1 buffer length
160	(A0)	CHARACTER	4		*	Reserved for 64 bit
164	(A4)	ADDRESS	4		CUNBOPRM_Wrk2_Buf_Ptr	Work2 buffer pointer
168	(A8)	UNSIGNED	4		CUNBOPRM_Wrk2_Buf_ALET	Work2 buffer ALET
172	(AC)	UNSIGNED	4		CUNBOPRM_Wrk2_Buf_Len	Work2 buffer length
176	(80)	CHARACTER	4		*	Reserved for 64 bit
180	(B4)	ADDRESS	4		CUNBOPRM_DDA_Buf_Ptr	Dynamic data area pointer
184	(B8)	UNSIGNED	4		CUNBOPRM_DDA_Buf_ALET	Dynamic data area ALET
188	(BC)	UNSIGNED	4		CUNBOPRM_DDA_Buf_Len	Dynamic data area length as defined by constant CUNBOPRM_DDA_Req.
192	(C0)	BITSTRING	1		CUNBOPRM_Flag1	FLAG Byte 1 set by caller
		1...			CUNBOPRM_Inv_Handle	Invalid handle action: 0=TERMINATE WITH ERROR. 1=GET NEW HANDLE AND CONT.
		.1...			CUNBOPRM_Get_New_Handle	Get a new handle 0=Get/Use a handle and continue with the service 1=Get handle and return to the caller
		..1.			CUNBOPRM_Page_Fix	Page Fixing: 0=System storage management (default). 1=Page Fixing.
193	(C1)	CHARACTER	1		*	Reserved
194	(C2)	BITSTRING	2		CUNBOPRM_Mask	Collation Mask

Collation

Table 19. Mapping of parameters in HLASM for collation AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
194	(C2)	BITSTRING	1		CUNBOPRM_Mask1	
		111.			CUNBOPRM_Variable_Opt	Where: 0=Shifted 1=Blanked 10=Non Blanked 11=Shift Trimmed and Reserved
		...1			CUNBOPRM_Cmp_Order	Where: 0=Forward 1=Backward (French)
		 1...			CUNBOPRM_Skey_Opt	Where: 0=No get sort key 1=Get sort key
		111			CUNBOPRM_Norm_Type	Normalization form 000=No Apply Norm. 001=Apply NFD 010=Apply NFC 011=Apply NFKD 100=Apply NFKC
195	(C3)	BITSTRING	1		CUNBOPRM_Mask2	
		1...			CUNBOPRM_GenSKey_and_Cmp	Make binary comparison (CUNBOPRM_RESULT) if and only if, CUNBOPRM_SKey_Opt is ON 0 - Do not perform binary comparison (Default) 1 - Perform binary comparison
196	(C4)	UNSIGNED	4		CUNBOPRM_Result	Comparison result: -1 if String1 < String2 0 if String1 = String2 1 if String1 > String2
200	(C8)	CHARACTER	8	WORD	CUNBOPRM_RC_RS	Return/reason code
200	(C8)	UNSIGNED	4		CUNBOPRM_Return_Code	Return code
204	(CC)	UNSIGNED	4		CUNBOPRM_Reason_Code	Reason code
208	(D0)	UNSIGNED	1		CUNBOPRM_UCA_Ver	Unicode Standard Version
209	(D1)	CHARACTER	2		*	Reserved
211	(D3)	CHARACTER	2		CUNBOPRM_Case_Options	Case Options
211	(D3)	UNSIGNED	1		CUNBOPRM_Case_First	Where: 0 - Default 1 - Upper First 10- Lower First
212	(D4)	BITSTRING	1		CUNBOPRM_Case_Options_Flags	Case Options

Table 19. Mapping of parameters in HLASM for collation AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
		1... ..			CUNBOPRM_Case_Level	Where: 0 - Default 1 - Primary Level will ignore accent but not case
213	(D5)	BITSTRING	1		CUNBOPRM_Special	Special chars considerations
		1... ..			CUNBOPRM_Hiragana	Distinguish between Japanese Hiragana and Katakana characters. 0 - Default 1 - Conform to the Japanese JIS X 4061 standard with Primary Level
214	(D6)	CHARACTER	2		*	Reserved
216	(D8)	UNSIGNED	4		CUNBOPRM_Var_Top	Variable Top - UTF16BE
220	(DC)	CHARACTER	32		CUNBOPRM_Locale	Locale Input (II_CC_Variant)
		CHARACTER	2		CUNBOPRM_Locale_II	Locale Language
		CHARACTER	1		*	Underscore
		CHARACTER	2		CUNBOPRM_Locale_CC	Locale Country/Region
		CHARACTER	1		*	Underscore
		CHARACTER	26		CUNBOPRM_Locale_Variant	Locale Variant
252	(FC)	CHARACTER	64		CUNBOPRM_Collation_Keyword	Collation parameters set - short form
316	(13C)	CHARACTER	44		CUNBOPRM_DSName	Collation rules DS Name
360	(168)	CHARACTER	4		*	Reserved
364	(16C)	CHARACTER	8		CUNBOPRM_Collation_Rules_File	File Name
372	(174)	CHARACTER	6		CUNBOPRM_Collation_Rules_Vol	Volume
378	(17A)	CHARACTER	2		*	Reserved
380	(17C)		0		CUNBOPRM_End	End of CUNBOPRM

Description of parameters in area CUNBOPRM

CUNBOPRM_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUNLOCOL using the constant CUNBOPRM_Ver which is supplied by the interface definition file CUNBOIDF.

In order to exploit new Collation features (new UCA versions UCA400R1, UCA410 and tailoring features), CUNBOPRM_Version must be set with CUNBOPRM_Ver2 (Collation parameter area version 2). For backward compatibility purposes, the default value is CUNBOPRM_Ver.

CUNBOPRM_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUNLOCOL using the constant CUNBOPRM_Len which is supplied by the interface definition file CUNBOIDF.

CUNBOPRM_Src1_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string of Unicode characters to be processed. No write operations are done in this field. The string has the length specified in the CUNBOPRM_Src1_Buf_Len parameter.

Note: Source buffer pointed by CUNBOPRM_Src1_Buf_Ptr must contain UTF-16 BE character format only. Otherwise, Collation Service will cause unpredictable results.

CUNBOPRM_Src1_Buf_ALET - set by caller

Specifies the ALET to be used if the source 1 buffer addressed by CUNBOPRM_Src1_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUNBOPRM_Src1_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUNBOPRM_Src1_Buf_Ptr, to be collated.

CUNBOPRM_Src2_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string of Unicode characters to be processed. No write operations are done in this field. The string has the length specified in the CUNBOPRM_Src2_Buf_Len parameter.

Note: Source buffer pointed by CUNBOPRM_Src2_Buf_Ptr must contain UTF-16 BE characters format only. Otherwise, Collation Service will cause unpredictable results. The UTF-16 BE character structure depends on the Unicode Standard Version specified at CUNBOPRM_UCA_Ver (The default is CUNBOPRM_UCA301) or CUNBOPRM_Collation_Keyword.

CUNBOPRM_Src2_Buf_ALET - set by caller

Specifies the ALET to be used if the source 2 buffer addressed by CUNBOPRM_Src2_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUNBOPRM_Src2_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUNBOPRM_Src2_Buf_Ptr, to be collated.

CUNBOPRM_Targ1_Buf_Ptr - set by caller, updated by service

This variable has two primary functions:

1. Binary comparison - If you need to do a comparison, you must specify two strings (to do a logical comparison). For this reason, CUNBOPRM_Targ1_Buf_Ptr needs to specify the beginning address and its related fields (CUNBOPRM_Targ1_Buf_ALET and CUNBOPRM_Targ1_Buf_Len) .
2. Sort key vector generation - If you need to generate a sort key vector, and you choose to set the CUNBOPRM_Src1_Buf_Ptr, you also need to set up its relative values (CUNBOPRM_Src1_Buf_ALET and CUNBOPRM_Src1_Buf_Len).

In both cases, it is important that you to set up this field correctly. For more information, see “Target buffer length considerations” on page 151 and “Sort key vector format” on page 149.

CUNBOPRM_Targ1_Buf_ALET - set by caller

Specifies the ALET to be used if the target 1 buffer addressed by CUNBOPRM_Targ1_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUNBOPRM_Targ1_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the target buffer addressed by CUNBOPRM_Targ1_Buf_Ptr. Certain conditions apply, dependent upon the collation level and the need for a sort key vector. See “Target buffer length considerations” on page 151 for more information.

CUNBOPRM_Targ2_Buf_Ptr - set by caller, updated by service

This variable has two primary functions:

1. Binary comparison - If you need to do a comparison, you must specify two strings (to do a logical comparison). For this reason, CUNBOPRM_Targ2_Buf_Ptr needs to specify the beginning address and its related fields (CUNBOPRM_Targ2_Buf_ALET and CUNBOPRM_Targ2_Buf_Len) .
2. Sort key vector generation - If you need to generate a sort key vector, and you choose to set the CUNBOPRM_Src2_Buf_Ptr, you also need to set up its relative values (CUNBOPRM_Src2_Buf_ALET and CUNBOPRM_Src2_Buf_Len).

In both cases, it is important that you to set up this field correctly. For more information, see “Target buffer length considerations” on page 151 and “Sort key vector format” on page 149.

CUNBOPRM_Targ2_Buf_ALET - set by caller

Specifies the ALET to be used if the target 2 buffer addressed by CUNBOPRM_Targ2_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUNBOPRM_Targ2_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the target buffer addressed by CUNBOPRM_Targ2_Buf_Ptr. Certain conditions apply, dependent upon the collation level and the need for a sort key vector. See “Target buffer length considerations” on page 151 for more information.

CUNBOPRM_Coll_Handle - set by caller, updated by service

Specifies the handle to the collation tables. If the handle is present, it will be used, otherwise a new handle will be returned in CUNBOPRM_Coll_Handle. Subsequent calls to stub routine CUNLOCOL, requesting the same collation properties, will be faster because then the handle is used and CUNBOPRM_Coll_Type does not need to be recomputed.

Note: For the first call to stub routine CUNLOCOL, CUNBOPRM_Coll_Handle must be set to binary zero X'00'.

CUNBOPRM_Coll_Level - set by caller

Specifies the collation level as defined by the following constants (defined in the interface definition file CUNBOIDF):

- CUNBOPRM_PRIMARY
- CUNBOPRM_SECONDARY
- CUNBOPRM_TERTIARY
- CUNBOPRM_QUATERNARY
- CUNBOPRM_QUINARY (Supported by UCA400R1 and higher)
- CUNBOPRM_IDENTICAL (Supported by UCA400R1 and higher)

Note:

1. CUNBOPRM_QUINARY and CUNBOPRM_IDENTICAL have exactly the same behavior and were added to cover multiple naming conventions for those Collation Levels.
2. Collation Levels are also named as "Collation Strength". See CUNBOPRM_Collation_Keyword field description.

CUNBOPRM_Wrk1_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string addressed by CUNBOPRM_Wrk1_Buf_Ptr. This variable is mainly used for internal purposes; however, it must always be set. See "Work buffer length considerations" on page 150 for more information.

CUNBOPRM_Wrk1_Buf_ALET - set by caller, updated by service

Specifies the ALET to be used if the work 1 buffer addressed by CUNBOPRM_Wrk1_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUNBOPRM_Wrk1_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work 1 buffer addressed by CUNBOPRM_Wrk1_Buf_Ptr. The length addressed will depend on the collation rules, including the collation level. See "Work buffer length considerations" on page 150 for more information.

CUNBOPRM_Wrk2_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string addressed by CUNBOPRM_Wrk2_Buf_Ptr. This variable is mainly used for internal purposes; however, it must always be set. See "Work buffer length considerations" on page 150 for more information.

CUNBOPRM_Wrk2_Buf_ALET - set by caller, updated by service

Specifies the ALET to be used if the work 2 buffer addressed by CUNBOPRM_Wrk2_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUNBOPRM_Wrk2_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work 2 buffer addressed by CUNBOPRM_Wrk2_Buf_Ptr. The length addressed will depend on the collation rules, including the collation level. See "Work buffer length considerations" on page 150 for more information.

CUNBOPRM_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that collation needs internally as a dynamic data area.

Note: CUNBOPRM_DDA_Buf_Ptr must be double-word boundary.

CUNBOPRM_DDA_Buf_ALET - set by caller

Specifies the ALET to be used if the dynamic data area addressed by CUNBOPRM_DDA_Buf_Ptr resides in a different address or data space. If not the primary address, the default value is 0.

CUNBOPRM_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUNBOPRM_DDA_Buf_Ptr. The required length is defined by constant CUNBOPRM_DDA_Req, which is provided in the interface definition file (CUNBOIDF).

CUNBOPRM_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUNBOPRM_Inv_Handle
x1xx xxxx	CUNBOPRM_Get_New_Handle
xx1x xxxx	CUNBOPRM_Page_Fix

CUNBOPRM_Inv_Handle

Specifies the action to be taken when the collation handle is invalid.

- **0:** Indicates that the collation is to be terminated with an error.
- **1:** Indicates that the collation is to be done with a new handle created by the collation service and put into CUNBOPRM_Coll_Handle.

CUNBOPRM_Get_New_Handle

Specifies the action to be taken with the new collation handle.

- **0:** Get and use the new handle and continue with the service.
- **1:** Get the new handle and return to the caller.

CUNBOPRM_Page_Fix

Specifies the action to be taken for the page fixing:

- **0:** Indicates use of system storage management (default).
- **1:** Indicates use of page fixing.

Note: CUNBOPRM_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUNBOPRM_Mask - set by caller

This parameter is two bytes in length, and together with CUNBOPRM_Coll_Level defines the collation rules. The default value is MASK_DEFAULT.

The following table shows the format and description of the sub fields.

Table 20. Collation mask sub fields descriptions

Sub fields	Description
CUNBOPRM_Variable_Opt	This sub field specifies if operations with variable collation elements must be performed. The options are: 0 - Shifted (SHIFTED) 1 - Blanked (BLANKED) 2 - Non-Ignored (NIGNORED) 3 - Shift-Trimmed (STRIMMED) 4 - No Variable Behavior (NAVARIABLECE)
CUNBOPRM_Cmp_Order	This sub field specifies following comparison orders: 0 - Forward (FORWARD) (Default) 1 - Backward (BACKWARD) (French behavior)
CUNBOPRM_SKey_Opt	This sub field specifies either a comparison or sort key: 0 - No get sort key (SKOFF) and perform binary comparison. 1 - Get sort key (SKON) and do not perform binary comparison.

Table 20. Collation mask sub fields descriptions (continued)

Sub fields	Description
CUNBOPRM_Norm_Type	This sub field specifies the normalization form according to the following values: 0 - No apply normalization (NNORM) (Default) 1 - Apply NFD (NFD) 2 - Apply NFC (NFC) 3 - Apply NFKD (NFKD) 4 - Apply NFKC (NFKC)
CUNBOPRM_GenSKey_and_Cmp	Perform Binary comparison when Sort Key is also requested. 0 - Do not perform binary comparison (default) 1 - perform binary comparison Note: This bit flag will be meaningful if the following flags are set: • CUNBOPRM_Version = CUNBOPRM_Ver2 • CUNBOPRM_SKey_Opt = SKON • CUNBOPRM_UCA_Ver = CUNBOPRM_UCA400R1 (or higher) Collation version 3.0.1, was able to generate either: • Perform Binary comparisons or • Generate Sort Key But not both. From UCA400R1 and higher, its possible to generate sort key and perform binary comparison at the same time.

CUNBOPRM_RESULT - updated by Service

Specifies the result of the binary comparison (between CUNBOPRM_Src1_Buf_Ptr and CUNBOPRM_Src2_Buf_Ptr).

The results can be evaluated according to the following values:

```
-1 if CUNBOPRM_Src1_Buf_Ptr < CUNBOPRM_Src2_Buf_Ptr
0 if CUNBOPRM_Src1_Buf_Ptr = CUNBOPRM_Src2_Buf_Ptr
1 if CUNBOPRM_Src1_Buf_Ptr > CUNBOPRM_Src2_Buf_Ptr
```

CUNBOPRM_RC_RS - set by service

A structure that can be used to access CUNBOPRM_Return_Code and CUNBOPRM_Reason_Code as one unit.

CUNBOPRM_Return_Code - set by service

Specifies the return code.

CUNBOPRM_Reason_Code - set by service

Specifies the reason code.

CUNBOPRM_UCA_VER - set by caller

Specifies the Unicode Collation Algorithm version (UCA) which also makes reference to the specific Unicode Standard character suite.

Note: This field will be referenced if Collation Parameter Area is set as CUNBOPRM_Version = CUNBOPRM_Ver2, otherwise its content will be ignored.

CUNBOPRM_Case_Options - set by caller

Specifies CASE options.

CUNBOPRM_Case_First - set by caller

Specifies whether upper case characters collate before lower case characters or not:

- 0 - Default (default value will depend on Locale. Most of the locales use Lower First as default.)
- 1 - Upper First
- 2 - Lower First

CUNBOPRM_Case_Options_Flags - set by caller

Setting CUNBOPRM_Case_Level to ON and CUNBOPRM_Coll_Level = CUNBOPRM_PRIMARY will ignore accent but not case:

- 0 - Default
- 1 - Ignore accent but not under primary collation

Note: Those fields will be referenced if Collation Parameter Area is set as CUNBOPRM_Version = CUNBOPRM_Ver2 and CUNBOPRM_UCA_VER is set to CUNBOPRM_UCA400R1 or CUNBOPRM_UCA401, otherwise its content will be ignored.

CUNBOPRM_Special - set by caller**CUNBOPRM_Hiragana - set by caller**

Specifies whether to distinguish between Japanese Hiragana and Katakana characters.

- 0 - Do not distinguish (default)
- 1 - Conform to the Japanese JIS X 4061 standard and use the CUNBOPRM_Coll_Level = CUNBOPRM_QUATERNARY collation.

Note: This field will be referenced if Collation Parameter Area is set as CUNBOPRM_Version = CUNBOPRM_Ver2 and CUNBOPRM_UCA_VER is set to CUNBOPRM_UCA400R1 or CUNBOPRM_UCA401, otherwise its content will be ignored.

CUNBOPRM_Var_Top - set by caller

Specifies the "highest" character (in UCA order) weight that is to be considered ignorable. The Variable Top attribute is only meaningful if the CUNBOPRM_Variable_Opt attribute is not set to Non-Ignored (NIGNORED). In such case, it controls which characters count as ignorable.

For example, if callers want white-space to be ignorable but not any visible characters, they would use the value CUNBOPRM_Var_Top = X'0020' (space). All characters of the same primary weight are equivalent, so CUNBOPRM_Var_Top=X'3000' (ideographic space) has the same effect as CUNBOPRM_Var_Top =X'0020'.

Note:

1. All valid Code Points must be under UTF-16 format.
2. Those fields will be referenced if Collation Parameter Area is set as CUNBOPRM_Version = CUNBOPRM_Ver2 and CUNBOPRM_UCA_VER is set to CUNBOPRM_UCA400R1 or CUNBOPRM_UCA410, otherwise its content will be ignored.

CUNBOPRM_Locale - set by caller

Specifies a locale, where specific Collation Rules will modify any of the default Unicode Collation tables specified (UCA400R1 or UCA410 - UCA301 does not support customization) and then Collation will behave according to those rules. Locales are set when you specify the following fields:

CUNBOPRM_Locale_Language - set by caller

Specify a language for desired locale.

CUNBOPRM_Locale_Region - set by caller

Specify a region for desired locale.

CUNBOPRM_Locale_Variant - set by caller

Specify a variant for desired locale.

Note:

1. For supported Locales settings (Language/Region/Variant), see Appendix F, "Locales support," on page 471.
2. If there is no Locale information, UCA version will be set as default without any change.
3. Those fields will be referenced if Collation Parameter Area is set as CUNBOPRM_Version = CUNBOPRM_Ver2 and CUNBOPRM_UCA_VER is set to CUNBOPRM_UCA400R1 or CUNBOPRM_UCA401, otherwise its content will be ignored.

Unicode Locales repository data set name SYS1.SCUNLOCL contains a set of locales documented in Appendix F, "Locales support," on page 471. All of those locales contain a section for Collation rules.

Users might want to copy locales and modify them as needed and then provide the locale name in CUNBOPRM_Locale sub-fields. Then you have to provide CUNBOPRM_DSName and CUNBOPRM_Collation_Rules_Vol in case that you want to load the locales with the Unicode dynamic capabilities. If that locale (modified by the users) is already loaded in the Unicode environment, there is no need to set data set and volume information.

The following example (CUNENUSX) shows how a locale looks like:

```
*****
* Licensed Materials - Property of IBM                               *
*                                                                      *
* "Restricted Materials of IBM"                                       *
*                                                                      *
* 5694-A01                                                            *
*                                                                      *
* (C) Copyright IBM Corp. 2006                                       *
*                                                                      *
* Status = HUN7730                                                    *
*                                                                      *
*****

<version $revision: 1.19 $ = default>
<collation>
  <rules>
    &\u0061\u0065
    <<\u00E6
```

```

        <<<\u00C6
    </rules>
</collation>
</version $revision: 1.19 $>

```

For further information about Locales, see Appendix F, “Locales support,” on page 471.

For further information about Collation rules syntax, see CUNBOPRM_Collation_Rules_File field description.

From Appendix F, “Locales support,” on page 471 the value shown in Column 2 for the Collation API field CUNBOPRM_Collation_Keyword is used for “short path”. Based on that field values for locales purpose, the following table shows some examples about how to get equivalencies between “short path” and “long path” settings.

Table 21. Equivalencies between short path and long path locale settings

CUNBOPRM_Collation_Keyword	CUNBOPRM_Locale_Language	CUNBOPRM_Locale_Region	CUNBOPRM_Locale_Variant
LAF	AF		
LAR_RBH	AR	BH	
LDE_RAT_VPREEURO	DE	AT	PREEURO
LZH_VPINYIN	ZH		PINYIN
LEN_RUS_VPOSIX	EN	US	POSIX

Locales information for CUNBOPRM_Collation_Keyword has the following prefixes:

- Lxx - For Language
- Ryy - For Region
- Vzz - For Variant

For CUNBOPRM_Locale_Language, CUNBOPRM_Locale_Region and CUNBOPRM_Locale_Variant, you can use exactly the same values but without the prefixes L, R or V.

Note: IBM does not recommend using CUNBOPRM_Locale directly, instead of that, use sub-fields CUNBOPRM_Locale_Language, CUNBOPRM_Locale_Region or CUNBOPRM_Locale_Variant.

CUNBOPRM_Collation_Keyword - set by caller

Specifies the “short path” settings form compatible with International Components for Unicode (ICU). IBM suggests you use this field instead of the “long path” settings for new Collation callers for UCA400R1 and UCA410 versions in the Collation API. This field can be set according the following table:

Collation

Table 22. Collation keywords descriptions

Attribute Name	Key	Possible Values	Description
Locale	L R V	<locale>	Provide a specific locale for collation rules which are in SYS1.SCUNLOCL repository. For Locales supported, see Appendix F, "Locales support," on page 471. Where "Attribute Name" has the following format: Lxx_Ryy_Vzz, where: • L means language • R means region • V means variant Example: UCA400R1_LSV (Swedish) "Kyper" < "Köpfe" For long path equivalent setting, see CUNBOPRM_Locale description.
Strength	S	1, 2, 3, 4, I, D	The Strength attribute determines whether accents or case are taken into account when collating or matching text (In UCA this is named Collation Levels. See CUNBOPRM_Coll_Level description). Example: UCA400R1_S1 role = Role = rôle UCA400R1_S2 role = Role < rôle UCA400R1_S3 role < Role < rôle For long path equivalent setting, see CUNBOPRM_Coll_Level description.
Case_Level	K	X, O, D	The Case Level attribute is used when ignoring accents but not case. In such case, set Strength to Primary, and Case_Level to On. In most locales, this setting is Off by default. Example: UCA400R1_S1_KX role = Role = rôle UCA400R1_S1_KO role = rôle < Role For long path equivalent setting, see CUNBOPRM_Case_Level description.
Case_First	C	X, L, U, D	The Case First attribute is used to control whether uppercase letters come before lowercase letters or vice versa in the absence of other differences in the strings. The possible values are Upper Case First (U) and Lower Case First (L), plus the standard Default and Off. There is almost no difference between the Off and Lower Case First options in terms of results, so typically users will not use Lower Case First but only Off or Upper Case First. Example: UCA400R1_CX or UCA400R1_CL "china" < "China" < "denmark" < "Denmark" UCA400R1_CU "China" < "china" < "Denmark" < "denmark" For long path equivalent setting, see CUNBOPRM_Case_First description.

Table 22. Collation keywords descriptions (continued)

Attribute Name	Key	Possible Values	Description
Alternate	A	N, S, D	The Alternate attribute is used to control the handling of the so-called variable characters in the UCA: white-space, punctuation and symbols. If Alternate is set to Non-Ignorable (N), then differences among these characters are of the same importance as differences among letters. If Alternate is set to Shifted (S), then these characters are of only minor importance. The Shifted value is often used in combination with Strength set to Quaternary. In such case, white-space, punctuation, and symbols are considered when comparing strings, but only if all other aspects of the strings (base letters, accents, and case) are identical. If Alternate is not set to Shifted, then there is no difference between a Strength of 3 and a Strength of 4. For more information and examples, see Variable_Weighting in the UCA. The reason the Alternate values are not simply On and Off is that additional Alternate values may be added in the future. The UCA option Blanked is expressed with Strength set to 3, and Alternate set to Shifted. Example: <pre>UCA400R1_S3_AN di Silva < Di Silva < diSilva < U.S.A. < USA UCA400R1_S3_AS di Silva = diSilva < Di Silva < U.S.A. = USA UCA400R1_S4_AS di Silva < diSilva < Di Silva < U.S.A. < USA</pre> For long path equivalent setting, see CUNBOPRM_Variable_Opt description.
Variable_Top	T	<hex digits>	The Variable Top attribute is only meaningful if the Alternate attribute is not set to Non-Ignorable. In such a case, it controls which characters count as ignorable. The string value specifies the "highest" character (in UCA order) weight that is to be considered ignorable. Thus, for example, if a user wanted white-space to be ignorable, but not any visible characters, then s/he would use the value Variable Top="u0020" (space). All characters of the same primary weight are equivalent, so Variable Top="u3000" (ideographic space) has the same effect as Variable_Top="u0020". Example: <pre>UCA400R1_S3_AN di Silva < diSilva < U.S.A. < USA UCA400R1_S3_AS di Silva = diSilva < U.S.A. = USA UCA400R1_S3_AS_T0020 di Silva = diSilva < U.S.A. = USA</pre> For long path equivalent setting, see CUNBOPRM_Var_Top description.
Normalization Checking	N	X, O, D	The Normalization setting determines whether text is thoroughly normalized or not in comparison (see also CUN4BOPR_Norm_Type). Example: <pre>UCA400R1_NX ä= a + ĩ% < ä+ ĩ% < i+ ĩ% UCA400R1_NO ä= a + ĩ% < ä+ ĩ% < i+ ĩ%</pre> For long path equivalent setting, see CUNBOPRM_Norm_Type description.
French	F	X, O, D	The French sort strings with different accents from the back of the string. This attribute is automatically set to On for the French locales and a few others. Users normally would not need to explicitly set this attribute. There is a string comparison performance cost when it is set On, but sort key length is not affected (see also CUN4BOPR_Cmp_Order). Example: <pre>UCA400R1_FX cote < coté< côte < cõtê UCA400R1_F0 cote < côte< coté < cõtê</pre> For long path equivalent setting, see CUNBOPRM_Cmp_Order description.

Collation

Table 22. Collation keywords descriptions (continued)

Attribute Name	Key	Possible Values	Description
Hiragana	H	X, O, D	Compatibility with JIS x 4061 requires the introduction of an additional level to distinguish Hiragana and Katakana characters. If compatibility with that standard is required, then this attribute should be set On, and the strength set to Quaternary. This will affect sort key length and string comparison string comparison performance. Example: UCA400R1_HX_S4 M0...= -å< M0†= -0æ UCA400R1_HO_S4 M0...< -å< M0†< -0æ For long path equivalent setting, see CUNBOPRM_Hiragana description.

Valid values for collation keywords are listed in the following table:

Table 23. Valid values for collation keywords

Value	Abbreviation
Default	D
On	O
Off	X
Primary	1
Secondary	2
Tertiary	3
Quaternary	4
Identical	I
Shifted	S
Non-Ignorable	N
Lower-First	L
Upper-First	U

These abbreviations allow a 'short path settings' specification of a set of collation options, such as "UCA400R1_AS_LSV_S2", which can be used to specify that the desired options are: UCA version 4.0.1; ignore spaces, punctuation and symbols; use Swedish linguistic conventions; compare case-insensitively.

A number of attribute values are common across different attributes; these include Default (abbreviated as D), On (O), and Off (X).

This form is compatible with ICU 3.2, however, the content of this short-set form fields is mutually exclusively from current collation configuration fields (long path settings), which means that this field will be the first one to be analyzed prior current collation fields content sets.

Note:

All collation keywords sets must start with either of the following Collation versions followed by desired sets:

- * UCA400R1_...
- * UCA410_...

If there is an invalid Keyword or invalid keyword value, Collation will return RC8/RS24 (CUN_RC_USER_ERR/CUN_RS_INVALID_COLLATION_KEYWORD_VALUES). If some of the keywords appear more than once, RC8/RS31 will be returned (CUN_RC_USER_ERR/CUN_RS_OVERLAYING_COLLATION_KEYWORD).

CUNBOPRM_DSName - set by caller

Specifies the name of the alternative data set from where the rules are to be loaded. It enables callers to load Locales from non-official Unicode repository (SYS1.SCUNLOCL) or load User Collation Rules Files from private data spaces as well (see CUNBOPRM_Collation_Rules_File).

CUNBOPRM_Collation_Rules_File - set by caller

Specifies member name where the alternative collation rules are. You can use User Collation Rules (UCR) for full Collation customization environment. Those files can be considered as a variation of Collation Rules or Locales since both UCR and Locales follow exactly the same collation syntax.

Collation rules can be redefined using the following symbols:

Table 24. Collation rule symbols

Symbol	Example	Description
<	\u0061<\u0062	Identifies a primary (base letter) difference between "a" and "b"
<<	\u0061<<\u00E4	Signifies a secondary (accent) difference between "a" and "ä"
<<<	\u0061<<<\u0041	Identifies a tertiary difference between "a" and "A"
=	x = y	Signifies no difference between "x" and "y". Note: X means CP x and Y means CP Y (x,y are not chars but CPs)
&	&Z	These rules will be relative to this letter, but will not affect the position of Z itself. Note: Z means CP Z (Z is not char but a CP)
/	æ/e	Expansion. Add the collation element for 'e' to the collation element for æ. After a reset "&æ << æ" is equivalent to "&a << æ/e".
	alb	Prefix processing. If 'b' is encountered and it follows 'a', output the appropriate collation element. If 'b' follows any other letter, output the normal collation element for 'b'. Collation element for 'a' is not affected.

Also the following tags might be part of the Collation syntax rules (default values are in BOLD and italic) as an easier way to set collation behavior:

Table 25. Collation syntax rules

Option	Example	Description
... ..	See CUNBOPRM_Locale parameter description field.	Describes the start/end block of sets for a locale. X.x and default denotes a locale revision/version, however, Locales versions are not meaningful at this time.
... ..	Refer to your default Unicode locales repository SYS1.SCUNLOCL and look for CUNAF locale.	Describes the start/end block of sets for a locale, where no revision and version are required, because default UCA rules are part of this locale.
... ..	See this table below .	Describes the start/end block of sets for a User Collation Rules (UCR). Default denotes an "UCR" version which is not meaningful at this time.
Alternate	[alternate non-ignorable] [alternate shifted]	Sets the default value for Alternate attribute. If set to shifted, variable code points will be ignored on the primary level.

Table 25. Collation syntax rules (continued)

Option	Example	Description
Backwards	[backwards 2]	Sets the default value for Backwards attribute. If set to on, secondary level will be reversed.
Variable top	& X < [variable top]	Sets the default value for Variable Top attribute. All the code points with primary strengths less than variable top will be considered variable.
Normalization Case Level	[normalization off] [normalization on]	Turns on or off the Normalization attribute . If set to on, a quick check and necessary normalization will be performed.
Case Level	[caseLevel off] [caseLevel on]	Turns on or off the Case Level attribute. If set to on a level consisting only of case characteristics will be inserted in front of tertiary level. To ignore accents but take cases into account, set strength to primary and case level to on.
Case First	[caseFirst off] [caseFirst upper] [caseFirst lower]	Sets the value for Case First attribute. If set to upper, causes upper case to sort before lower case. If set to lower, lower case will sort before upper case. Useful for locales that have already supported ordering but require different order of cases. Affects case and tertiary levels.
Strength	[strength 1] [strength 2] [strength 3] [strength 4] [strength 5] [strength l]	Sets the default strength attribute.
Hiragana	[hiraganaQ off] [hiraganaQ on]	Controls special treatment of Hiragana code points on quaternary level. If turned on, Hiragana code points will get lower values than all the other non-variable code points. Strength must be greater or equal than quaternary if you want this attribute to take effect. Set UCOE_HIRAGANAQ.
[before 1 2 3]	&[before 1] a<?<à<?<á?	Enables users to order characters before a given character. In UCA 3.0, the example is equivalent to &?<?<à<?<á? (?= \u3029, Hangzhou numeral nine) * and makes accented 'a' letters sort before 'a'. Accents are often used to indicate the intonations in Pinyin. In this case, the non-accented letters sort after the accented letters.
[last non ignorable]	&[last non ignorable]<\u4E9C	Defines a list of CP's which will be positioned right after [last non-ignorable] CP.
[last regular]	&[last regular]<\u4E9C	Equivalent as [last non-ignorable]
[suppressContractions [FromCP-ToCP]]	&[suppressContractions [\u0400-\u045F]]	Suppress all contraction defined in a range defined by FromCP - ToCP. After this rule, all of them will be treated as Normal CP's.
[last secondary ignorable]	&[last secondary ignorable]<<\u0020	All CP's after [last secondary ignorable] will be placed after last secondary ignorable CP.

The following is an example which can be used as UCR files:

```

*****
* Owner: My Name *
* Prof Description: User Collation Rules profile sample *
* *
* *
* *
*

```



```

*
*
*
*
*****
<version $UCR$ = default>
<collation>
  <rules>
    [strength 1]           * Collation Settings ...
    [alternate non-ignorable]
    [backwards 2]
    [normalization on]
    [caseLevel on]
    [caseFirst off]
    [hiraganaQ off]
    &\u0061\u0065           * Modifying CPs
      <<\u00E6
      <<<\u00C6
    &\u0062<\u0061
  </rules>
</collation>
</version $UCR$ = default>

```

For Collation Rules Files or locales files consider the following:

- Use the asterisk "*" as a comment line, starting at column 1.
- Whatever collation settings must be specified inside of the tags <rules> ... </rules>.
- All collation tags and values are key sensitive. Use exact same tags and UTF-16 CP format as specified on this section .
- As part of code points, use the following UTF-16, that is, \u0061. "\u" denotes a UTF-16 CP.
- Blanks are not allowed after each one of the following symbols:
 - =\u
 - <\u
 - <<\u
 - <<<\u
 - ^\u

For this new collation implementation (tailoring for UCA400R1 and higher - not available for UCA301), there are two ways to perform collation settings in the Collation API. You must follow the following order in case that more than one is specified in the Collation API.

1. Short path - This setting is based on the contents of CUNBOPRM_Collation_Keyword
For example, "UCA400R1_LEN_RUS_VPOSIX"
2. Long path - This setting is used when some of the following fields are set and values are followed according to its order in the following list:
 - CUNBOPRM_Coll_Level
 - CUNBOPRM_Variable_Opt
 - CUNBOPRM_Cmp_Order
 - CUNBOPRM_SKey_Opt
 - CUNBOPRM_Norm_Type
 - CUNBOPRM_Case_First
 - CUNBOPRM_Case_Level
 - CUNBOPRM_Hiragana

- CUNBOPRM_Var_Top
- CUNBOPRM_Locale_Language, CUNBOPRM_Locale_Region or CUNBOPRM_Locale_Variant
- CUNBOPRM_Collation_Rules_File

Note: For long path settings, collation API fields like CUNBOPRM_Coll_Level, CUNBOPRM_Variable ... CUNBOPRM_Var_Top override any Collation settings on Locales (CUNBOPRM_Locale) or UCR (CUNBOPRM_Collation_Rules_File).

CUNBOPRM_Collation_Rules_Vol - set by caller

Specifies the volume for data set specified by CUNBOPRM_DSName.

Mapping of constants for AMODE (31)

For HLASM, you can set up the parameter area CUNBOPRM with a group of constants that are provided in the interface definition file for collation (CUNB0IDF).

```
* *****
* *                               CUNBOPRM_Mask Constants                               *
* * xxx- ---- CUNBOPRM_Mask field into CUNBOPRM                                     *
* * Where CUNBOPRM_Mask is a sub-structure into CUNBOPRM structure *
* *****
*
*
MASK_DEFAULT EQU X'E0'           Non-ApplyVCE + Not Backward +
*
* *****!
* *                                           !
* * * NSK + Not Norm                               !
* * *                                           !
* *****!
*
*
* *****
* * xxx- ---- *
* *           Where xxx is CUNBOPRM_Variable_Opt field *
* *****
*
*
SHIFTED EQU X'00'      Shift
BLANKED EQU X'20'      Blanked
NIGNORED EQU X'40'     Not-Ignored
STRIMMED EQU X'60'     Shift-Trimmed
NAVARIABLECE EQU X'E0' No Variable CE
*
* *****
* * ---X ---- *
* *           Where ---x is CUNBOPRM_Cmp_Order field *
* *****
*
*
BACKWARD EQU X'10'     Backward Order
FORWARD EQU X'00'      Frowand Order
*
* *****
* * ---- X--- *
* *           Where x is CUNBOPRM_SKey_Opt field *
* *****
*
*
SKOFF EQU X'00'        Sort Key OFF
SKON EQU X'08'         Sort Key ON
```

```

*
* *****
* * ---- -XXX                                     *
* *           Where xxx is CUNBOPRM_Norm_Type field       *
* *****
*
*
NNORM    EQU    X'00'        Not Norm
NFD      EQU    X'01'        Can Decomp
NFC      EQU    X'02'        Can Comp
NFKD     EQU    X'03'        Compat Dec
NFKC     EQU    X'04'        Compat Com
*
* *****
* *           CUNBOPRM_Flag1 Constants                     *
* * xy-- ---- CUNBOPRM_Flag1 field into CUNBOPRM         *
* *           Where x--- ---- CUNBOPRM_Inv_Handle; and    *
* *           -y-- ---- CUNBOPRM_Get_New_Handle          *
* *****
*
*
FLAG1_DEFAULT    EQU X'00'        Flag1 Default
INV_HANDLE_ON    EQU X'80'        Get Handle ON
GET_NEW_HANDLE_ON EQU X'40'        Get_New_Handle ON
*
* *****
* *           Other Collation Constants                     *
* *****
*
*                               * Maximum Collation Level
*
MAXVALIDLEVEL EQU 5        Available
ALTERNATE_NON_IGNOREABLE EQU B'0'
ALTERNATE_SHIFTED EQU B'1'
BACKWARDS_OFF EQU B'0'
BACKWARDS_ON EQU B'1'
NORMALIZATION_OFF EQU B'0'
NORMALIZATION_ON EQU B'1'
CASELEVEL_OFF EQU B'0'
CASELEVEL_ON EQU B'1'
CASEFIRST_OFF EQU 0
CASEFIRST_UPPER EQU 1
CASEFIRST_LOWER EQU 2
STRENGTH_I EQU 5
STRENGTH_1 EQU 1
STRENGTH_2 EQU 2
STRENGTH_3 EQU 3
STRENGTH_4 EQU 4
STRENGTH_5 EQU 5
HIRAGANAQ_OFF EQU B'0'
HIRAGANAQ_ON EQU B'1'
CUNBOPRM_LEN EQU *-CUNBOPRM
*
* *****
* *           Constant to initialize CUNBOPRM_Version.     *
* *****
*
*
CUNBOPRM_VER EQU 1
CUNBOPRM_VER2 EQU 2
*
* *****
* *           Constant defining the required Dynamic Data Area (DDA) size. *
* *****
*
*
CUNBOPRM_DDA_BUF_MIN EQU 800 DDa min Buf

```

Collation

```

CUNBOPRM_DDA_REQ EQU 4096 Required Dynamic data area size.
*
* *****
* *                               Constant UCA Versions                               *
* *****
*
*
CUNBOPRM_UCAEMPTY EQU 0
CUNBOPRM_UCA301 EQU 1
CUNBOPRM_UCA400R1 EQU 2
CUNBOPRM_UCA410 EQU 3
*
* *****
* *                               CUNBOPRM_Coll_Level Constants                               *
* *****
*
*
CUNBOPRM_IDENTICAL EQU 5 Identical
CUNBOPRM_PRIMARY EQU 1 First Level
CUNBOPRM_SECONDARY EQU 2 Second Level
CUNBOPRM_TERTIARY EQU 3 Third Level
CUNBOPRM_QUATERNARY EQU 4 Fourth Level
CUNBOPRM_QUINARY EQU 5 Fifth Level

```

Note: IBM suggests you use "OR" operations to add collation rules. If you add any value directly, the field will lose the previous designation.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas is supplied by the interface definition file CUN4BOID. This file is shipped in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that may be necessary.

Table 26. Mapping of parameters in HLASM for collation AMODE (64)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	428	DWORD	CUN4BOPR	Parameter Area
0	(0)	UNSIGNED	4		CUN4BOPR_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUN4BOPR_Length	Parameter area Length
8	(8)	ADDRESS	8		CUN4BOPR_Src1_Buf_Ptr	Source1 buffer pointer
16	(10)	CHARACTER	4		*	Reserved for 64 bit
20	(14)	UNSIGNED	4		CUN4BOPR_Src1_Buf_ALET	Source1 buffer ALET
24	(18)	UNSIGNED	8		CUN4BOPR_Src1_Buf_Len	Source1 buffer length
32	(20)	ADDRESS	8		CUN4BOPR_Src2_Buf_Ptr	Source2 buffer pointer
40	(28)	CHARACTER	4		*	Reserved for 64 bit
44	(2C)	UNSIGNED	4		CUN4BOPR_Src2_Buf_ALET	Source2 buffer ALET
48	(30)	UNSIGNED	8		CUN4BOPR_Src2_Buf_Len	Source2 buffer length
56	(38)	ADDRESS	8		CUN4BOPR_Targ1_Buf_Ptr	Target1 buffer pointer
64	(40)	CHARACTER	4		*	Reserved for 64 bit
68	(44)	UNSIGNED	4		CUN4BOPR_Targ1_Buf_ALET	Target1 buffer ALET
72	(48)	UNSIGNED	8		CUN4BOPR_Targ1_Buf_Len	Target1 buffer length
80	(50)	ADDRESS	8		CUN4BOPR_Targ2_Buf_Ptr	Target2 buffer pointer
88	(58)	CHARACTER	4		*	Reserved for 64 bit

Table 26. Mapping of parameters in HLASM for collation AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
92	(5C)	UNSIGNED	4		CUN4BOPR_Targ2_Buf_ALET	Target2 buffer ALET
96	(60)	UNSIGNED	8		CUN4BOPR_Targ2_Buf_Len	Target2 buffer length
104	(68)	CHARACTER	64	DWORD	CUN4BOPR_Coll_Handle	Collation handle
168	(A8)	CHARACTER	1		CUN4BOPR_Coll_Level	Collation level
169	(A9)	CHARACTER	7		*	Reserved
176	(B0)	ADDRESS	8		CUN4BOPR_Wrk1_Buf_Ptr	Work1 buffer pointer
184	(B8)	CHARACTER	4		*	Reserved for 64 bit
188	(BC)	UNSIGNED	4		CUN4BOPR_Wrk1_Buf_ALET	Work1 buffer ALET
192	(C0)	UNSIGNED	8		CUN4BOPR_Wrk1_Buf_Len	Work1 buffer length
200	(C8)	ADDRESS	8		CUN4BOPR_Wrk2_Buf_Ptr	Work2 buffer pointer
208	(D0)	CHARACTER	4		*	Reserved for 64 bit
212	(D4)	UNSIGNED	4		CUN4BOPR_Wrk2_Buf_ALET	Work2 buffer ALET
216	(D8)	UNSIGNED	8		CUN4BOPR_Wrk2_Buf_Len	Work2 buffer length
224	(E0)	ADDRESS	8		CUN4BOPR_DDA_Buf_Ptr	Dynamic data area pointer
232	(E8)	UNSIGNED	4		CUN4BOPR_DDA_Buf_ALET	Dynamic data area ALET
236	(EC)	UNSIGNED	4		CUN4BOPR_DDA_Buf_Len	Dynamic data area length as defined by constant CUN4BOPR_DDA_Req.
240	(F0)	BITSTRING	1		CUN4BOPR_Flag1	FLAG Byte 1 set by caller
		1...			CUN4BOPR_Inv_Handle	Invalid handle action: 0=TERMINATE WITH ERROR. 1=GET NEW HANDLE AND CONT.
		.1...			CUN4BOPR_Get_New_Handle	Get a new handle 0=Get/Use a handle and continue with the service 1=Get handle and return to the caller
		..1.			CUN4BOPR_Page_Fix	Page Fixing: 0=System storage management (default). 1=Page Fixing.
241	(F1)	CHARACTER	1		*	Reserved
242	(F2)	BITSTRING	2		CUN4BOPR_Mask	Collation Mask
242	(F2)	BITSTRING	1		CUN4BOPR_Mask1	

Collation

Table 26. Mapping of parameters in HLASM for collation AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
		111.			CUN4BOPR_Variable_Opt	Where: 0=Shifted 1=Blanked 10=Non Blanked 11=Shift Trimmed and Reserved
		...1			CUN4BOPR_Cmp_Order	Where: 0=Forward 1=Backward (French)
		 1...			CUN4BOPR_SKey_Opt	Where: 0=No get sort key 1=Get sort key
		111			CUN4BOPR_Norm_Type	Normalization form 000=No Apply Norm. 001=Apply NFD 010=Apply NFC 011=Apply NFKD 100=Apply NFKC
243	(F3)	BITSTRING	1		CUN4BOPR_Mask2	
		1...			CUN4BOPR_GenSKey_and_Cmp	Make binary comparison (CUNBOPRM_RESULT) if and only if, CUN4BOPR_SKey_Opt is ON 0 - Do not perform binary comparison (Default) 1 - Perform binary comparison
244	(F4)	UNSIGNED	4		CUN4BOPR_Result	Comparison result: -1 if String1 < String2 0 if String1 = String2 1 if String1 > String2
248	(F8)	CHARACTER	8	WORD	CUN4BOPR_RC_RS	Return/reason code
248	(F8)	UNSIGNED	4		CUN4BOPR_Return_Code	Return code
252	(FC)	UNSIGNED	4		CUN4BOPR_Reason_Code	Reason code
256	(100)	UNSIGNED	1		CUN4BOPR_UCA_Ver	Unicode Standard Version
257	(101)	CHARACTER	2		*	Reserved
259	(103)	CHARACTER	2		CUN4BOPR_Case_Options	Case Options
259	(103)	UNSIGNED	1		CUN4BOPR_Case_First	Where: 0 - Default 1 - Upper First 10- Lower First
260	(104)	BITSTRING	1		CUN4BOPR_Case_Options_Flags	Case Options

Table 26. Mapping of parameters in HLASM for collation AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
		1... ..			CUN4BOPR_Case_Level	Where: 0 - Default 1 - Primary Level will ignore accent but not case
261	(105)	BITSTRING	1		CUN4BOPR_Special	Special chars considerations
		1... ..			CUN4BOPR_Hiragana	Distinguish between Japanese Hiragana and Katakana characters. 0 - Default 1 - Conform to the Japanese JIS X 4061 standard with Primary Level
262	(106)	CHARACTER	2		*	Reserved
264	(108)	UNSIGNED	4		CUN4BOPRM_Var_Top	Variable Top - UTF16BE
268	(10C)	CHARACTER	32		CUN4BOPRM_Locale	Locale Input (II_CC_Variant)
		CHARACTER	2		CUN4BOPRM_Locale_II	Locale Language
		CHARACTER	1		*	Underscore
		CHARACTER	2		CUN4BOPRM_Locale_CC	Locale Country/Region
		CHARACTER	1		*	Underscore
		CHARACTER	26		CUN4BOPRM_Locale_Variant	Locale Variant
300	(12C)	CHARACTER	64		CUN4BOPRM_Collation_Keyword	Collation parameters set - short form
364	(16C)	CHARACTER	44		CUN4BOPRM_DSName	Collation rules DSName
408	(198)	CHARACTER	4		*	Reserved
412	(19C)	CHARACTER	8		CUN4BOPRM_Collation_Rules_File	File Name
420	(1A4)	CHARACTER	6		CUN4BOPRM_Collation_Rules_Vol	Volume
426	(1AA)	CHARACTER	2		*	Reserved
428	(1AC)		0		CUN4BOPR_End	End of CUN4BOPR

Description of parameters in area CUN4BOPR

CUN4BOPR_Version - set by caller

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUN4LCOL using the constant CUN4BOPR_Ver which is supplied by the interface definition file CUN4BOLD.

In order to exploit new Collation features (new UCA versions UCA400R1, UCA410 and tailoring features), CUN4BOPR_Version must be set with CUN4BOPR_Ver2 (Collation parameter area version 2). For backward compatibility purposes, the default value is CUN4BOPR_Ver.

CUN4BOPR_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUN4LCOL using the constant CUN4BOPR_Len which is supplied by the interface definition file CUN4BOLD.

CUN4BOPR_Src1_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string of Unicode characters to be processed. No write operations are done in this field. The string has the length specified in the CUN4BOPR_Src1_Buf_Len parameter.

Note: Source buffer pointed by CUN4BOPR_Src1_Buf_Ptr must contain UTF-16 BE characters format only. Otherwise, Collation Service will cause unpredictable results.

CUN4BOPR_Src1_Buf_ALET - set by caller

Specifies the ALET to be used if the source 1 buffer addressed by CUN4BOPR_Src1_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUN4BOPR_Src1_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUN4BOPR_Src1_Buf_Ptr, to be collated.

CUN4BOPR_Src2_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string of Unicode characters to be processed. No write operations are done in this field. The string has the length specified in the CUN4BOPR_Src2_Buf_Len parameter.

Note: Source buffer pointed to by CUN4BOPR_Src2_Buf_Ptr must contain UTF-16 BE character format only. Otherwise, Collation Service will cause unpredictable results.

CUN4BOPR_Src2_Buf_ALET - set by caller

Specifies the ALET to be used if the source 2 buffer addressed by CUN4BOPR_Src2_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUN4BOPR_Src2_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUN4BOPR_Src2_Buf_Ptr, to be collated.

CUN4BOPR_Targ1_Buf_Ptr - set by caller, updated by service

This variable has two primary functions:

1. Binary comparison - If you need to do a comparison, you must specify two strings (to do a logical comparison). For this reason, CUN4BOPR_Targ1_Buf_Ptr needs to specify the beginning address and its related fields (CUN4BOPR_Targ1_Buf_ALET and CUN4BOPR_Targ1_Buf_Len).
2. Sort key vector generation - If you need to generate a sort key vector, and you choose to set the CUN4BOPR_Src1_Buf_Ptr, you also need to set up its relative values (CUN4BOPR_Src1_Buf_ALET and CUN4BOPR_Src1_Buf_Len).

In both cases, it is important that you to set up this field correctly. For more information, see “Target buffer length considerations” on page 151 and “Sort key vector format” on page 149.

CUN4BOPR_Targ1_Buf_ALET - set by caller

Specifies the ALET to be used if the target 1 buffer addressed by

CUN4BOPR_Targ1_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUN4BOPR_Targ1_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the target buffer addressed by CUN4BOPR_Targ1_Buf_Ptr. Certain conditions apply, dependent upon the collation level and the need for a sort key vector. See “Target buffer length considerations” on page 151 for more information.

CUN4BOPR_Targ2_Buf_Ptr - set by caller, updated by service

This variable has two primary functions:

1. Binary comparison - If you need to do a comparison, you must specify two strings (to do a logical comparison). For this reason, CUN4BOPR_Targ2_Buf_Ptr needs to specify the beginning address and its related fields (CUN4BOPR_Targ2_Buf_ALET and CUN4BOPR_Targ2_Buf_Len).
2. Sort key vector generation - If you need to generate a sort key vector, and you choose to set the CUN4BOPR_Src2_Buf_Ptr, you also need to set up its relative values (CUN4BOPR_Src2_Buf_ALET and CUN4BOPR_Src2_Buf_Len).

In both cases, it is important that you to set up this field correctly. For more information, see “Target buffer length considerations” on page 151 and “Sort key vector format” on page 149.

CUN4BOPR_Targ2_Buf_ALET - set by caller

Specifies the ALET to be used if the target 2 buffer addressed by CUN4BOPR_Targ2_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUN4BOPR_Targ2_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the target buffer addressed by CUN4BOPR_Targ2_Buf_Ptr. Certain conditions apply, dependent upon the collation level and the need for a sort key vector. See “Target buffer length considerations” on page 151 for more information.

CUN4BOPR_Coll_Handle - set by caller, updated by service

Specifies the handle to the collation tables. If the handle is present, it will be used, otherwise a new handle will be returned in CUN4BOPR_Coll_Handle. Subsequent calls to stub routine CUN4LCOL, requesting the same collation properties, will be faster because then the handle is used and CUN4BOPR_Coll_Type does not need to be recomputed.

Note: For the first call to stub routine CUN4LCOL, CUN4BOPR_Coll_Handle must be set to binary zero X'00'.

CUN4BOPR_Coll_Level - set by caller

Specifies the collation level as defined by the following constants (defined in the interface definition file CUN4BOLD):

- CUN4BOPR_PRIMARY
- CUN4BOPR_SECONDARY
- CUN4BOPR_TERTIARY
- CUN4BOPR_QUATERNARY
- CUN4BOPR_QUINARY (Supported by UCA400R1 and higher)
- CUN4BOPR_IDENTICAL (Supported by UCA400R1 and higher)

Note:

1. CUN4BOPR_QUINARY and CUN4BOPR_IDENTICAL have exactly the same behavior and were added to cover multiple naming conventions for those Collation Levels.
2. Collation Levels are also named as "Collation Strength". See CUN4BOPR_Collation_Keyword field description.

CUN4BOPR_Wrk1_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string addressed by CUN4BOPR_Wrk1_Buf_Ptr. This variable is mainly used for internal purposes; however, it must always be set. See "Work buffer length considerations" on page 150 for more information.

CUN4BOPR_Wrk1_Buf_ALET - set by caller, updated by service

Specifies the ALET to be used if the work 1 buffer addressed by CUN4BOPR_Wrk1_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUN4BOPR_Wrk1_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work 1 buffer addressed by CUN4BOPR_Wrk1_Buf_Ptr. The length addressed will depend on the collation rules, including the collation level. See "Work buffer length considerations" on page 150 for more information.

CUN4BOPR_Wrk2_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the string addressed by CUN4BOPR_Wrk2_Buf_Ptr. This variable is mainly used for internal purposes; however, it must always be set. See "Work buffer length considerations" on page 150 for more information.

CUN4BOPR_Wrk2_Buf_ALET - set by caller, updated by service

Specifies the ALET to be used if the work 2 buffer addressed by CUN4BOPR_Wrk2_Buf_Ptr resides in a different data space. If not the primary address, the default value is 0.

CUN4BOPR_Wrk2_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work 2 buffer addressed by CUN4BOPR_Wrk2_Buf_Ptr. The length addressed will depend on the collation rules, including the collation level. See "Work buffer length considerations" on page 150 for more information.

CUN4BOPR_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that collation needs internally as a dynamic data area.

Note: CUN4BOPR_DDA_Buf_Ptr must be double-word boundary.

CUN4BOPR_DDA_Buf_ALET - set by caller

Specifies the ALET to be used if the dynamic data area addressed by CUN4BOPR_DDA_Buf_Ptr resides in a different address or data space. If not the primary address, the default value is 0.

CUN4BOPR_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUN4BOPR_DDA_Buf_Ptr. The required length is defined by constant CUN4BOPR_DDA_Req, which is provided in the interface definition file (CUN4BOLD).

CUN4BOPR_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUN4BOPR_Inv_Handle
x1xx xxxx	CUN4BOPR_Get_New_Handle
xx1x xxxx	CUN4BOPR_Page_Fix

CUN4BOPR_Inv_Handle

Specifies the action to be taken when the collation handle is invalid.

- **0:** Indicates that the collation is to be terminated with an error.
- **1:** Indicates that the collation is to be done with a new handle created by the collation service and put into CUN4BOPR_Coll_Handle.

CUN4BOPR_Get_New_Handle

Specifies the action to be taken with the new collation handle.

- **0:** Get and use the new handle and continue with the service.
- **1:** Get the new handle and return to the caller.

CUN4BOPR_Page_Fix

Specifies the action to be taken for the page fixing:

- **0:** Indicates use of system storage management (default).
- **1:** Indicates use of page fixing.

Note: CUN4BOPR_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUN4BOPR_Mask - set by caller

This parameter is two bytes in length, and together with CUN4BOPR_Coll_Level defines the collation rules. The default value is MASK_DEFAULT.

The following table shows the format and description of the sub fields.

Table 27. Collation mask sub fields descriptions

Sub fields	Description
CUN4BOPR_Variable_Opt	This sub field specifies if operations with variable collation elements must be performed. The options are: 0 - Shifted (SHIFTED) 1 - Blanked (BLANKED) 2 - Non-Ignored (NIGNORED) 3 - Shift-Trimmed (STRIMMED) 4 - No Variable Behavior (NAVARIABLECE)
CUN4BOPR_Cmp_Order	This sub field specifies following comparison orders: 0 - Forward (FORWARD) (Default) 1 - Backward (BACKWARD) (French behavior)
CUN4BOPR_SKey_Opt	This sub field specifies either a comparison or sort key: 0 - No get sort key (SKOFF) and perform binary comparison.(Default) 1 - Get sort key (SKON) and do not perform binary comparison.

Table 27. Collation mask sub fields descriptions (continued)

Sub fields	Description
CUN4BOPR_Norm_Type	This sub field specifies the normalization form according to the following values: 0 - No apply normalization (NNORM) (Default) 1 - Apply NFD (NFD) 2 - Apply NFC (NFC) 3 - Apply NFKD (NFKD) 4 - Apply NFKC (NFKC)
CUN4BOPR_GenSKey_and_Cmp	Perform Binary comparison when Sort Key is also requested. 0 - Do not perform binary comparison (default) 1 - perform binary comparison Note: This bit flag will be meaningful if the following flags are set: • CUN4BOPR_Version = CUN4BOPR_Ver2 • CUN4BOPR_SKey_Opt = SKON • CUN4BOPR_UCA_Ver = CUN4BOPR_UCA400R1 (or higher) Collation version 3.0.1, was able to generate either: • Perform Binary comparisons or • Generate Sort Key But not both. From UCA400R1 and higher, its possible to generate sort key and perform binary comparison at the same time.

CUN4BOPR_RESULT - updated by service

Specifies the result of the binary comparison (between CUN4BOPR_Src1_Buf_Ptr and CUN4BOPR_Src2_Buf_Ptr).

The results can be evaluated according to the following values:

```
-1 if CUN4BOPR_Src1_Buf_Ptr < CUN4BOPR_Src2_Buf_Ptr
0  if CUN4BOPR_Src1_Buf_Ptr = CUN4BOPR_Src2_Buf_Ptr
1  if CUN4BOPR_Src1_Buf_Ptr > CUN4BOPR_Src2_Buf_Ptr
```

CUN4BOPR_RC_RS - set by service

A structure that can be used to access CUN4BOPR_Return_Code and CUN4BOPR_Reason_Code as one unit.

CUN4BOPR_Return_Code - set by service

Specifies the return code.

CUN4BOPR_Reason_Code - set by service

Specifies the reason code.

CUN4BOPR_UCA_VER - set by caller

Specifies the Unicode Collation Algorithm version (UCA) which also makes reference to the specific Unicode Standard character suite.

Note: This field will be referenced if Collation Parameter Area is set as CUN4BOPR_Version = CUN4BOPR_Ver2, otherwise its content will be ignored.

CUN4BOPR_Case_Options - set by caller

Specifies CASE options.

CUN4BOPR_Case_First - set by caller

Specifies whether upper case characters collate before lower case characters or not:

- 0 - Default (default value will depend on Locale. Most of the locales use Lower First as default.)
- 1 - Upper First
- 2 - Lower First

CUN4BOPR_Case_Options_Flags - set by caller

Setting CUN4BOPR_Case_Level to ON and CUN4BOPR_Coll_Level = CUN4BOPR_PRIMARY will ignore accent but not case:

- 0 - Default
- 1 - Ignore accent but not under primary collation

Note: Those fields will be referenced if Collation Parameter Area is set as CUN4BOPR_Version = CUN4BOPR_Ver2 and CUN4BOPR_UCA_VER is set to CUN4BOPR_UCA400R1 or CUN4BOPR_UCA410, otherwise its content will be ignored.

CUN4BOPR_Special - set by caller**CUN4BOPR_Hiragana - set by caller**

Specifies whether to distinguish between Japanese Hiragana and Katakana characters.

- 0 - Do not distinguish (default)
- 1 - Conform to the Japanese JIS X 4061 standard and use the CUN4BOPR_Coll_Level = CUN4BOPR_QUATERNARY collation.

Note: This field will be referenced if Collation Parameter Area is set as CUN4BOPR_Version = CUN4BOPR_Ver2 and CUN4BOPR_UCA_VER is set to CUN4BOPR_UCA400R1 or CUN4BOPR_UCA410, otherwise its content will be ignored.

CUN4BOPR_Var_Top - set by caller

Specifies the "highest" character (in UCA order) weight that is to be considered ignorable. The Variable Top attribute is only meaningful if the CUN4BOPR_Variable_Opt attribute is not set to Non-Ignored (NIGNORED). In such case, it controls which characters count as ignorable.

For example, if callers want white-space to be ignorable but not any visible characters, they would use the value CUN4BOPR_Var_Top=X'0020' (space). All characters of the same primary weight are equivalent, so CUN4BOPR_Var_Top=X'3000' (ideographic space) has the same effect as CUNBOPRM_Var_Top=X'0020'.

Note:

1. All valid Code Points must be under UTF-16 format.
2. Those fields will be referenced if Collation Parameter Area is set as CUN4BOPR_Version = CUN4BOPR_Ver2 and CUN4BOPR_UCA_VER is set to CUN4BOPR_UCA400R1 or CUN4BOPR_UCA410, otherwise its content will be ignored.

CUN4BOPR_Locale - set by caller

Specifies a locale, where specific Collation Rules will modify any of the default Unicode Collation tables specified (UCA400R1 or UCA410 - UCA301 does not support customization) and then Collation will behave according to those rules. Locales are set when you specify the following fields:

CUN4BOPR_Locale_Language - set by caller

Specify a language for desired locale.

CUN4BOPR_Locale_Region - set by caller

Specify a region for desired locale.

CUN4BOPR_Locale_Variant - set by caller

Specify a variant for desired locale.

Note:

1. For supported Locales settings (Language/Region/Variant), see Appendix F, "Locales support," on page 471.
2. If there is no Locale information, UCA version will be set as default without any change.
3. Those fields will be referenced if Collation Parameter Area is set as CUN4BOPR_Version = CUN4BOPR_Ver2 and CUN4BOPR_UCA_VER is set to CUN4BOPR_UCA400R1 or CUN4BOPR_UCA410, otherwise its content will be ignored.

Unicode Locales repository data set name SYS1.SCUNLOCL contains a set of locales documented in Appendix F, "Locales support," on page 471. All of those locales contain a section for Collation rules.

Users might want to copy locales and modify them as needed and then provide the locale name in CUN4BOPR_Locale sub-fields. Then you have to provide CUN4BOPR_DSName and CUN4BOPR_Collation_Rules_Vol in case that you want to load the locales with the Unicode dynamic capabilities. If that locale (modified by the users) is already loaded in the Unicode environment, there is no need to set data set and volume information.

The following example (CUNENUSX) shows how a locale looks like:

```
*****
* Licensed Materials - Property of IBM                               *
*                                                                      *
* "Restricted Materials of IBM"                                       *
*                                                                      *
* 5694-A01                                                            *
*                                                                      *
* (C) Copyright IBM Corp. 2006                                       *
*                                                                      *
* Status = HUN7730                                                    *
*                                                                      *
*****

<version $revision: 1.19 $ = default>
<collation>
  <rules>
    &\u0061\u0065
    <<\u00E6
    <<<\u00C6
  </rules>
</collation>
</version $revision: 1.19 $>
```

For further information about Locales, see Appendix F, “Locales support,” on page 471.

For further information about Collation rules syntax, see CUN4BOPR_Collation_Rules_File field description.

From Appendix F, “Locales support,” on page 471 the value shown in Column 2 for the Collation API field CUN4BOPR_Collation_Keyword is used for “short path”. Based on that field values for locales purpose, the following table shows some examples about how to get equivalencies between “short path” and “long path” settings.

Table 28. Equivalencies between short path and long path local settings

CUN4BOPR_Collation_Keyword	CUN4BOPR_Locale_Language	CUN4BOPR_Locale_Region	CUN4BOPR_Locale_Variant
LAF	AF		
LAR_RBH	AR	BH	
LDE_RAT_VPREURO	DE	AT	PREURO
LZH_VPINYIN	ZH		PINYIN
LEN_RUS_VPOSIX	EN	US	POSIX

Locales information for CUN4BOPR_Collation_Keyword has the following prefixes:

- Lxx - For Language
- Ryy - For Region
- Vzz - For Variant

For CUN4BOPR_Locale_Language, CUN4BOPR_Locale_Region and CUN4BOPR_Locale_Variant, you can use exactly the same values but without the prefixes L, R or V.

Note: IBM does not recommend to use CUN4BOPR_Locale directly, instead of that, use sub-fields CUN4BOPR_Locale_Language, CUN4BOPR_Locale_Region or CUN4BOPR_Locale_Variant.

CUN4BOPR_Collation_Keyword - set by caller

Specifies the “short path” settings form compatible with International Components for Unicode (ICU). IBM suggests you use this field instead of the “long path” settings for new Collation callers for UCA400R1 and UCA410 versions in the Collation API. This field can be set according the following table:

Table 29. Collation keywords descriptions

Attribute Name	Key	Possible Values	Description
Locale	L R V	<locale>	Provide a specific locale for collation rules which are in SYS1.SCUNLOCL repository. For Locales supported, see Appendix F, “Locales support,” on page 471. Where “Attribute Name” has the following format: Lxx_Ryy_Vzz, where: • L means language • R means region • V means variant Example: UCA400R1_LSV (Swedish) "Kypper" < "Köpfe" For long path equivalent setting, see CUNBOPRM_Locale description.

Collation

Table 29. Collation keywords descriptions (continued)

Attribute Name	Key	Possible Values	Description
Strength	S	1, 2, 3, 4, I, D	The Strength attribute determines whether accents or case are taken into account when collating or matching text (In UCA this is named Collation Levels. See CUNBOPRM_Coll_Level description). Example: <pre>UCA400R1_S1 rôle = Rôle = rôle UCA400R1_S2 rôle = Rôle < rôle UCA400R1_S3 rôle < Rôle < rôle</pre> For long path equivalent setting, see CUNBOPRM_Coll_Level description.
Case_Level	K	X, O, D	The Case Level attribute is used when ignoring accents but not case. In such case, set Strength to Primary, and Case_Level to On. In most locales, this setting is Off by default. Example: <pre>UCA400R1_S1_KX rôle = Rôle = rôle UCA400R1_S1_KO rôle = rôle < Rôle</pre> For long path equivalent setting, see CUNBOPRM_Case_Level description.
Case_First	C	X, L, U, D	The Case First attribute is used to control whether uppercase letters come before lowercase letters or vice versa in the absence of other differences in the strings. The possible values are Upper Case First (U) and Lower Case First (L), plus the standard Default and Off. There is almost no difference between the Off and Lower Case First options in terms of results, so typically users will not use Lower Case First but only Off or Upper Case First. Example: <pre>UCA400R1_CX or UCA400R1_CL "china" < "China" < "denmark" < "Denmark" UCA400R1_CU "China" < "china" < "Denmark" < "denmark"</pre> For long path equivalent setting, see CUNBOPRM_Case_First description.
Alternate	A	N, S, D	The Alternate attribute is used to control the handling of the so-called variable characters in the UCA: white-space, punctuation and symbols. If Alternate is set to Non-Ignorable (N), then differences among these characters are of the same importance as differences among letters. If Alternate is set to Shifted (S), then these characters are of only minor importance. The Shifted value is often used in combination with Strength set to Quaternary. In such case, white-space, punctuation, and symbols are considered when comparing strings, but only if all other aspects of the strings (base letters, accents, and case) are identical. If Alternate is not set to Shifted, then there is no difference between a Strength of 3 and a Strength of 4. For more information and examples, see Variable_Weighting in the UCA. The reason the Alternate values are not simply On and Off is that additional Alternate values may be added in the future. The UCA option Blanked is expressed with Strength set to 3, and Alternate set to Shifted. Example: <pre>UCA400R1_S3_AN di Silva < Di Silva < diSilva < U.S.A. < USA UCA400R1_S3_AS di Silva = diSilva < Di Silva < U.S.A. = USA UCA400R1_S4_AS di Silva < diSilva < Di Silva < U.S.A. < USA</pre> For long path equivalent setting, see CUNBOPRM_Variable_Opt description.

Table 29. Collation keywords descriptions (continued)

Attribute Name	Key	Possible Values	Description
Variable_Top	T	<hex digits>	The Variable Top attribute is only meaningful if the Alternate attribute is not set to Non-Ignorable. In such a case, it controls which characters count as ignorable. The string value specifies the "highest" character (in UCA order) weight that is to be considered ignorable. Thus, for example, if a user wanted white-space to be ignorable, but not any visible characters, then s/he would use the value Variable Top="u0020" (space). All characters of the same primary weight are equivalent, so Variable Top="u3000" (ideographic space) has the same effect as Variable_Top="u0020". Example: <pre>UCA400R1_S3_AN di Silva < diSilva < U.S.A. < USA UCA400R1_S3_AS di Silva = diSilva < U.S.A. = USA UCA400R1_S3_AS_T0020 di Silva = diSilva < U.S.A. = USA</pre> For long path equivalent setting, see CUNBOPRM_Var_Top description.
Normalization Checking	N	X, O, D	The Normalization setting determines whether text is thoroughly normalized or not in comparison (see also CUN4BOPR_Norm_Type). Example: <pre>UCA400R1_NX ä= a + ĩ% < ä+ ĩ% < i+ ĩ% UCA400R1_NO ä= a + ĩ% < ä+ ĩ% < i+ ĩ%</pre> For long path equivalent setting, see CUNBOPRM_Norm_Type description.
French	F	X, O, D	The French sort strings with different accents from the back of the string. This attribute is automatically set to On for the French locales and a few others. Users normally would not need to explicitly set this attribute. There is a string comparison performance cost when it is set On, but sort key length is not affected (see also CUN4BOPR_Cmp_Order). Example: <pre>UCA400R1_FX cote < coté< côte < cõtê UCA400R1_F0 cote < côte< coté < cõtê</pre> For long path equivalent setting, see CUNBOPRM_Cmp_Order description.
Hiragana	H	X, O, D	Compatibility with JIS x 4061 requires the introduction of an additional level to distinguish Hiragana and Katakana characters. If compatibility with that standard is required, then this attribute should be set On, and the strength set to Quaternary. This will affect sort key length and string comparison string comparison performance. Example: <pre>UCA400R1_HX_S4 M0...= -â< M0†= -0æ UCA400R1_HO_S4 M0...< -â< M0†< -0æ</pre> For long path equivalent setting, see CUNBOPRM_Hiragana description.

Valid values for collation keywords are listed in the following table:

Table 30. Valid values for collation keywords

Value	Abbreviation
Default	D
On	O
Off	X
Primary	1
Secondary	2
Tertiary	3
Quaternary	4

Table 30. Valid values for collation keywords (continued)

Value	Abbreviation
Identical	I
Shifted	S
Non-Ignorable	N
Lower-First	L
Upper-First	U

These abbreviations allow a 'short path settings' specification of a set of collation options, such as "UCA400R1_AS_LSV_S2", which can be used to specify that the desired options are: UCA version 4.0.1; ignore spaces, punctuation and symbols; use Swedish linguistic conventions; compare case-insensitively.

A number of attribute values are common across different attributes; these include Default (abbreviated as D), On (O), and Off (X).

This form is compatible with ICU 3.2, however, the content of this short-set form fields is mutually exclusively from current collation configuration fields (long path settings), which means that this field will be the first one to be analyzed prior current collation fields content sets.

Note:

All collation keywords sets must start with either of the following Collation versions followed by desired sets:

- * UCA400R1_...
- * UCA410_...

If there is an invalid Keyword or invalid keyword value, Collation will return RC8/RS24 (CUN_RC_USER_ERR/CUN_RS_INVALID_COLLATION_KEYWORD_VALUES). If some of the keywords appear more than once, RC8/RS31 will be returned (CUN_RC_USER_ERR/CUN_RS_OVERLAYING_COLLATION_KEYWORD).

CUN4BOPR_DSName - set by caller

Specifies the name of the alternative data set from where the rules are to be loaded. It enables callers to load Locales from non-official Unicode repository (SYS1.SCUNLOCL) or load User Collation Rules Files from private data spaces as well (see CUN4BOPR_Collation_Rules_File).

CUN4BOPR_Collation_Rules_File - set by caller

Specifies member name where the alternative collation rules are. You can use User Collation Rules (UCR) for full Collation customization environment. Those files can be considered as a variation of Collation Rules or Locales since both UCR and Locales follow exactly the same collation syntax.

Collation rules can be redefined using the following symbols:

Table 31. Collation rule symbols

Symbol	Example	Description
<	\u0061<\u0062	Identifies a primary (base letter) difference between "a" and "b"
<<	\u0061<<\u00E4	Signifies a secondary (accent) difference between "a" and "ä"

Table 31. Collation rule symbols (continued)

Symbol	Example	Description
<<<	\u0061<<<\u0041	Identifies a tertiary difference between "a" and "A"
=	x = y	Signifies no difference between "x" and "y". Note: X means CP x and Y means CP Y (x,y are not chars but CPs)
&	&Z	These rules will be relative to this letter, but will not affect the position of Z itself. Note: Z means CP Z (Z is not char but a CP)
/	æ/e	Expansion. Add the collation element for 'e' to the collation element for æ. After a reset "&ae << æ" is equivalent to "&a << æ/e".
l	alb	Prefix processing. If 'b' is encountered and it follows 'a', output the appropriate collation element. If 'b' follows any other letter, output the normal collation element for 'b'. Collation element for 'a' is not affected.

Also the following tags might be part of the Collation syntax rules (default values are in **BOLD** and *italic*) as an easier way to set collation behavior:

Table 32. Collation syntax rules

Option	Example	Description
... ..	See CUNBOPRM_Locale parameter description field.	Describes the start/end block of sets for a locale. X.x and default denotes a locale revision/version, however, Locales versions are not meaningful at this time.
... ..	Refer to your default Unicode locales repository SYS1.SCUNLOCL and look for CUNAF locale.	Describes the start/end block of sets for a locale, where no revision and version are required, because default UCA rules are part of this locale.
... ..	See this table below .	Describes the start/end block of sets for a User Collation Rules (UCR). Default denotes an "UCR" version which is not meaningful at this time.
Alternate	[alternate non-ignorable] [alternate shifted]	Sets the default value for Alternate attribute. If set to shifted, variable code points will be ignored on the primary level.
Backwards	[backwards 2]	Sets the default value for Backwards attribute. If set to on, secondary level will be reversed.
Variable top	& X < [variable top]	Sets the default value for Variable Top attribute. All the code points with primary strengths less than variable top will be considered variable.
Normalization Case Level	[normalization off] [normalization on]	Turns on or off the Normalization attribute . If set to on, a quick check and necessary normalization will be performed.
Case Level	[caseLevel off] [caseLevel on]	Turns on or off the Case Level attribute. If set to on a level consisting only of case characteristics will be inserted in front of tertiary level. To ignore accents but take cases into account, set strength to primary and case level to on.
Case First	[caseFirst off] [caseFirst upper] [caseFirst lower]	Sets the value for Case First attribute. If set to upper, causes upper case to sort before lower case. If set to lower, lower case will sort before upper case. Useful for locales that have already supported ordering but require different order of cases. Affects case and tertiary levels.

Table 32. Collation syntax rules (continued)

Option	Example	Description
Strength	[strength 1] [strength 2] [strength 3] [strength 4] [strength 5] [strength l]	Sets the default strength attribute.
Hiragana	[hiraganaQ off] [hiraganaQ on]	Controls special treatment of Hiragana code points on quaternary level. If turned on, Hiragana code points will get lower values than all the other non-variable code points. Strength must be greater or equal than quaternary if you want this attribute to take effect. Set UCOE_HIRAGANAQ.
[before 1 2 3]	&[before 1] a<?<à<?<á?	Enables users to order characters before a given character. In UCA 3.0, the example is equivalent to &?<?<à<?<á? (?= \u3029, Hangzhou numeral nine) * and makes accented 'a' letters sort before 'a'. Accents are often used to indicate the intonations in Pinyin. In this case, the non-accented letters sort after the accented letters.
[last non ignorable]	&[last non ignorable]<\u4E9C	Defines a list of CP's which will be positioned right after [last non-ignorable] CP.
[last regular]	&[last regular]<\u4E9C	Equivalent as [last non-ignorable]
[suppressContractions [FromCP-ToCP]]	&[suppressContractions [u0400-\u045F]]	Suppress all contraction defined in a range defined by FromCP - ToCP. After this rule, all of them will be treated as Normal CP's.
[last secondary ignorable]	&[last secondary ignorable]<<<\u0020	All CP's after [last secondary ignorable] will be placed after last secondary ignorable CP.

The following is an example which can be used as UCR files:

```

*****
* Owner: My Name *
* Prof Description: User Collation Rules profile sample *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
*****
<version $UCR$ = default>
<collation>
  <rules>
    [strength 1] * Collation Settings ...
    [alternate non-ignorable]
    [backwards 2]
    [normalization on]
    [caseLevel on]
    [caseFirst off]
    [hiraganaQ off]
    &\u0061\u0065 * Modifying CPs
    <<\u00E6
    <<<\u00C6
    &\u0062<\u0061
  </rules>
</collation>
</version $UCR$ = default>

```

For Collation Rules Files or locales files consider the following:

- Use the asterisk "*" as a comment line, starting at column 1.
- Whatever collation settings must be specified inside of the tags <rules> ... </rules>.
- All collation tags and values are key sensitive. Use exact same tags and UTF-16 CP format as specified on this section .
- As part of code points, use the following UTF-16, that is, \u0061. "\u" denotes a UTF-16 CP.
- Blanks are not allowed after each one of the following symbols:
 - =\u
 - <\u
 - <<\u
 - <<<\u
 - /\u

For this new collation implementation (tailoring for UCA400R1 and higher - not available for UCA301), there are two ways to perform collation settings in the Collation API. You must follow the following order in case that more than one is specified in the Collation API.

1. Short path - This setting is based on the contents of CUN4BOPR_Collation_Keyword
For example, "UCA400R1_LEN_RUS_VPOSIX"
2. Long path - This setting is used when some of the following fields are set and values are followed according to its order in the following list:
 - CUN4BOPR_Coll_Level
 - CUN4BOPR_Variable_Opt
 - CUN4BOPR_Cmp_Order
 - CUN4BOPR_SKey_Opt
 - CUN4BOPR_Norm_Type
 - CUN4BOPR_Case_First
 - CUN4BOPR_Case_Level
 - CUN4BOPR_Hiragana
 - CUN4BOPR_Var_Top
 - CUN4BOPR_Locale_Language, CUN4BOPR_Locale_Region or CUN4BOPR_Locale_Variant
 - CUN4BOPR_Collation_Rules_File

Note: For long path settings, collation API fields like CUN4BOPR_Coll_Level, CUN4BOPR_Variable ... CUN4BOPR_Var_Top override any Collation settings on Locales (CUN4BOPR_Locale) or UCR (CUN4BOPR_Collation_Rules_File).

CUN4BOPR_Collation_Rules_Vol - set by service

Specify the volume for data set specified by CUN4BOPR_DSName.

Mapping of constants for AMODE (64)

For HLASM, you can set up the parameter area (CUN4BOPR) with a group of constants that are provided in the interface definition file for collation (CUN4BOLD).

```

* *****
* *                               CUN4BOPR_Mask Constants                               *
* * xxx- ---- CUN4BOPR_Mask field into CUN4BOPR                                     *
* * Where CUN4BOPR_Mask is a sub-structure into CUN4BOPR structure *
* *****
*
* MASK_DEFAULT EQU X'E0'           Non-ApplyVCE + Not Backward +
*
* * *****!
* * *
* * * NSK + Not Norm *
* * *
* * *****!
*
*
* *****
* * XXX- ---- *
* * Where xxx is CUN4BOPR_Variable_Opt field *
* *****
*
*
* SHIFTED EQU X'00'      Shift
* BLANKED EQU X'20'      Blanked
* NIGNORED EQU X'40'     Not-Ignored
* STRIMMED EQU X'60'     Shift-Trimmed
* NAVARIABLECE EQU X'E0' No Variable CE
*
* *****
* * ---X ---- *
* * Where ---x is CUN4BOPR_Cmp_Order field *
* *****
*
*
* BACKWARD EQU X'10'      Backward Order
* FORWARD EQU X'00'      Forward Order
*
* *****
* * ---- X--- *
* * Where x is CUN4BOPR_SKey_Opt field *
* *****
*
*
* SKOFF EQU X'00'      Sort Key OFF
* SKON EQU X'08'      Sort Key ON
*
* *****
* * ---- -XXX *
* * Where xxx is CUN4BOPR_Norm_Type field *
* *****
*
*
* NNORM EQU X'00'      Not Norm
* NFD EQU X'01'      Can Decomp
* NFC EQU X'02'      Can Comp
* NFKD EQU X'03'      Compat Dec
* NFKC EQU X'04'      Compat Com
*
* *****
* *                               CUN4BOPR_Flag1 Constants                               *
* * xy-- ---- CUN4BOPR_Flag1 field into CUN4BOPR                                     *
* * Where x--- ---- CUN4BOPR_Inv_Handle; and *
* * -y-- ---- CUN4BOPR_Get_New_Handle *
* *****
*
*

```

```

FLAG1_DEFAULT      EQU X'00'      Flag1 Default
INV_HANDLE_ON      EQU X'80'      Get Handle ON
GET_NEW_HANDLE_ON  EQU X'40'      Get_New_Handle ON
*
* *****
* *                               Other Collation Constants                               *
* *****
*
*                               * Maximum Collation Level
*
MAXVALIDLEVEL EQU 4      Available
*
CUN4BOPR_DDA_BUF_MIN EQU 800 DDA min Buf
CUN4BOPR_DDA_REQ EQU 4096 Required Dynamic data area size.
*
* *****
* *                               CUN4BOPR_Coll_Level Constants                               *
* *****
*
*
CUN4BOPR_IDENTICAL EQU 0 Identical
CUN4BOPR_PRIMARY EQU 1 First Level
CUN4BOPR_SECONDARY EQU 2 Second Level
CUN4BOPR_TERTIARY EQU 3 Third Level
CUN4BOPR_QUATERNARY EQU 4 Fourth Level

```

Note: IBM suggests you use "OR" operations to add collation rules. If you add any value directly, the field will lose the previous designation.

Sort key vector format

The sort key, or sort key vector, is a collection of weights which come from the file allkeys.txt. This vector is stored in the target buffers of the parameter area, followed by two main restrictions:

- Sort key option ON (CUNBOPRM_SKey_Opt = SKON)
- The CUNBOPRM_SrcX_Buf_Ptr, with some valid addressed information (where X could be 1 or 2)

Also, the sort key vector has two principal variations:

1. Contents - depends on the CUNBOPRM_MASK, which can generate some different results according its combinations.
2. Size - defined by collation level specified in the CUNBOPRM_Coll_Level field, and by CUNBOPRM_Norm_Type, which is a sub field from the CUNBOPRM_MASK.

Consequently, the length of the sort key vector will depend on the number of Unicode characters set to the respective source (1 or 2), and the collation rules (CUNBOPRM_Coll_Level and CUNBOPRM_MASK).

The weights of the Unicode characters will be combined by level, then a separator must be inserted (X'0000') before the concatenated weight for the next level, and so on. This process is executed for as many collation levels as have been specified (1 to 4).

The size of the sort key vector is related to the collation level, as shown in the following table:

Table 33. Collation level weight length

Collation Level	Weight length in bytes
L1	2
L2	2
L3	1
L4	2

For any given Unicode character with a selected collation level, its collation sort key will be formed in the following format:

www0000xxxx0000yy0000zzzz

where:

www represents level one (two bytes)
 xxxx represents level two (two bytes)
 yy represents level three (one byte)
 zzzz represents level four (two bytes)

0000 represents the collation level separator (two bytes). For an example:

Unicode characters: FD3F,2495,FE30

Weight entries:

FD3F ; [*0287.0020.0002.FD3F]
 # ORNATE RIGHT PARENTHESIS
 2495 ; [.0858.0020.0004.2495] [.085B.0020.0004.2495] [*0241.0020.0004.2495]
 # NUMBER FOURTEEN
 FE30 ; [*0241.0020.0016.FE30] [*0241.0020.0016.FE30]
 # PRESENTATION FORM FOR VERTICAL TWO DOT

The collation options assumed are collation level=3, and variable_opt = ignored.

Sort key formed, would be:

02870858085B024102410241000000200020002000200020000020404041616

For UCA version UCA400R1 and higher, size of sort key is increased due to new infrastructure for tailoring purposes and also add support for surrogates as part of new Collation versions (UCA400R1 and UCA410). Even the size of the sort key per Code Point might have many variations according the settings. For target buffers size, see section on “Target buffer length considerations” on page 151.

Work buffer length considerations

The work buffer length has the same considerations for both 31-bit and 64-bit. There are two main considerations, both of them are related to the collation level you specify. Following are the two possibilities:

- Case 1 - CUNBOPRM_Coll_Level = 1, 2 or 3. For this case, you must consider at least twice the value of the source length (CUNBOPRM_SrcX_Buf_Len * 2), where X could be 1 or 2.
- Case 2 - CUNBOPRM_Coll_Level = 4. For this level, you must require at least three times the value of the source (SrcX_Buf_Len * 3), where X could be 1 or 2.

For UCA version UCA400R1 and higher, the following table shows the size of the work buffers for most common UTF-16BE Code Points:

Table 34. Size of the work buffers for UTF-16BE Code Points

Collation Level / Strength	Work Buffer length per Code Point in Source buffer
1	4 - Bytes
2	7 - Bytes
3	9 - Bytes
4	12 - Bytes
5	15 - Bytes

Note:

Most common UTF-16BE Code Points require 2-bytes in Source buffer. Non-normal CP's are expansions, contractions, surrogates, surrogates expansions and surrogates contractions.

IBM recommends allocating the same bytes for work buffer as for target buffer, see section on "Target buffer length considerations." If Collation returns with RC = CUN_RC_USER_ERR, RS = CUN_RS_WRK_EXHAUSTED by following this recommendation (Wrk Buffer Len = Target buffer length), it is recommended to multiply failed work buffer length by 2 and so on.

Target buffer length considerations

The target buffer length has the same considerations for both 31-bit and 64-bit. The following explains how you can set the size of the CUNBOPRM_TargX_Buf_Len parameter (where X could be 1 or 2).

1. Binary comparison - In this case, many combinations must be considered, due to the kind of normalization that has been specified. see Chapter 6, "Normalization," on page 77 for more information.
2. Sort key vector - the main use of the target buffer is to keep the sort key vector from CUNBOPRM_TargX_Buf_Ptr (where x could be 1 or 2). The size of this parameter is based upon several factors.

The following table shows a brief reference of recommended lengths for the various collation levels.

Table 35. Recommended target buffer lengths for collation

Collation Level	IBM recommended length
L1	Len1 = CUNBOPRM_SrcX_Buf_Len
L2	Len2 = CUNBOPRM_SrcX_Buf_Len * 2 + 2
L3	Len3 = (CUNBOPRM_SrcX_Buf_Len * 3) + 2
L4	Len4 = (CUNBOPRM_SrcX_Buf_Len * 4) + 2

For UCA version UCA400R1 and higher, the following table shows the size of the target buffers for most common UTF-16BE Code Points:

Table 36. Size of the target buffers for UTF-16BE Code Points

Collation Level / Strength	Target Buffer length per Code Point in Source buffer	Collation Separator size between intermediate Collation Levels
1	4 - Bytes	4 - Bytes

Collation

Table 36. Size of the target buffers for UTF-16BE Code Points (continued)

Collation Level / Strength	Target Buffer length per Code Point in Source buffer	Collation Separator size between intermediate Collation Levels
2	7 - Bytes	3 - Bytes
3	9 - Bytes	2 - Bytes
4	12 - Bytes	2 - Bytes
5	15 - Bytes	Not required

For Collation sort keys which live on target buffers, it is required to consider the Collation separator size.

Consider the following example:

Source Buffer Len = 4 (two UTF-16BE CP's
CP' on Src Buffer = Source Buffer Len / 2

Table 37. Target Buffer Formula

Collation Level / Strength	Target Buffer Formula
1	(CP' on Src Buffer * 4)
2	(CP' on Src Buffer * 4) + 4 (CP' on Src Buffer * 3)
3	(CP' on Src Buffer * 4) + 4 (CP' on Src Buffer * 3) + 3 (CP' on Src Buffer * 2)
4	(CP' on Src Buffer * 4) + 4 (CP' on Src Buffer * 3) + 3 (CP' on Src Buffer * 2) + 2 (CP' on Src Buffer * 3)
5 or I	(CP' on Src Buffer * 4) + 4 (CP' on Src Buffer * 3) + 3 (CP' on Src Buffer * 2) + 2 (CP' on Src Buffer * 3)

Note: For target buffers size when current work buffer length does not satisfy Collation requirements and returns with RC = CUN_RC_ERR, RS = CUN_RS_TARG_EXHAUSTED), it is recommended to multiply failed target buffer length by 2 and so on.

See “Sort key vector format” on page 149 for more information.

Sample programs

Sample programs for collation are provided in SYS1.SAMPLIB. The following table shows the AMODE and the API used (C/C++ or HLASM) in combination with long or short path settings.

Table 38. The AMODE and API (C/C++ or HLASM) in combination with long or short path settings

Program Name	AMODE 31-Bit	AMODE 64-Bit	Coll API C/C++	Coll API HLASM	UCA Version	Long Path	Short Path
CUNSOSMC	X		X		UCA301		
CUNSOSMA	X			X	UCA301		
CUN4SOSA		X		X			
CUN4SOSC		X	X				
CUNSO00C	X		X		UCA400R1	X	

Table 38. The AMODE and API (C/C++ or HLASM) in combination with long or short path settings (continued)

Program Name	AMODE 31-Bit	AMODE 64-Bit	Coll API C/C++	Coll API HLASM	UCA Version	Long Path	Short Path
CUNSO01C	X		X		UCA400R1		X
CUNSO02C		X	X		UCA400R1	X	
CUNSO03C		X	X		UCA400R1		X
CUNSO04A	X			X	UCA400R1	X	
CUNSO05A	X			X	UCA400R1		X
CUNSO06A		X		X	UCA400R1	X	
CUNSO07A		X		X	UCA400R1		X

End of Programming Interface information

Chapter 8. Bidi transformation

Programming Interface information

This chapter describes the programming required for the Bidi transformation service.

Bidi is also referred to as Unicode System Services for Bidi and character shaping services. The Bidi transformation service is called using a stub routine named CUNLBIDI for AMODE (31), and CUN4LBID for AMODE (64).

Bidi defines a minimal set of directional formatting codes to control the ordering of characters when rendered. This allows exact control of the display ordering for legible interchange and also ensures that plain text used for simple items like filenames or labels can always be correctly ordered for display.

This z/OS Unicode implementation meets some specifications described in the Unicode Standard Annex #9 "The bidirectional Algorithm" (For z/OS v1R8 Bidi only supports mirroring and character inversion). For further information about the Bidi and Character Shaping Service, see the Unicode Standard Annex #9 (<http://www.unicode.org/reports/tr9/>).

Bidi transformation services for Unicode provide two different ways to invoke them, with a new API and also for an ease of use, conversion character services now support a brand new technique "B" which makes the transformation on the output buffer but preserving the current behavior.

Calling Bidi transformation service

This section describes how to call the Bidi transformation and character shaping service.

The 31-bit caller has to provide:

- Source buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Target buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Work buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Source CCSID (4 byte)
- Target CCSID (4 byte)
- Flags

The 64-bit caller has to provide:

- Source buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Work buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Source CCSID (4 byte)
- Target CCSID (4 byte)
- Flags

Using the C interface

This topic describes the syntax in C for calling the stub routine **CUNLBIDI** or **CUN4LBID** (Bidi). Mapping of the parameter area is supplied by the header file `cunhc.h` listed in “Mapping of parameters in C.”

```
#include<cunhc.h>
#define SLEN 1024
#define WLEN 4096
#define TLEN 4096
.....
unsigned char Sourcebuffer [SLEN];
unsigned char Workbuffer   [WLEN];
unsigned char Targetbuffer [TLEN];

#ifdef _LP64      /* 64 bit */
CUN4BBPR myparm ={CUN4BBPR_DEFAULT};
#else            /* 31 bit */
CUNBBPRM myparm ={CUNBBPRM_DEFAULT};
#endif

Myparm.Src_Buf_Ptr  = Sourcebuffer;
Myparm.Wrk_Buf_Ptr  = Workbuffer;
myparm.Targ_Buf_Ptr = Targetbuffer;
Myparm.Src_Buf_Len  = SLEN;
Myparm.Wrk_Buf_Len  = WLEN;
myparm.Targ_Buf_Len = TLEN;
Myparm.ccsid_src     = 1200;
Myparm.ccsid_trt     = 425;

#ifdef _LP64      /* 31 bit */
CUNLBIDI(&myparm);
#else            /* 64 bit */
CUN4LBID(&myparm);
#endif

if((myparm.Return_Code != CUN_RC_OK).....
```

Mapping of parameters in C

A C header file is supplied (`cunhc.h`) that contains the function prototypes for the Bidi service. The following structure is used in the interface to the Bidi service.

31-bit mapping

```
typedef struct tag_CUNBBPRM{
    int  version;                /* Parameter Area Version */
    int  length;                 /* Parameter Area Length  */
    int  res1;                   /* Reserved for 64 bit     */
    void *Src_Buf_Ptr;           /* Pointer to Source       */
    int  res2;                   /* Reserved for 64 bit     */
    unsigned int Src_Buf_ALET;    /* ALET of source buffer   */
    int  res3;                   /* Reserved for 64 bit     */
    unsigned long Src_Buf_Len;    /* Length of source data   */
    int  res4;                   /* Reserved for 64 bit     */
    void *Targ_Buf_Ptr;          /* Pointer to Target       */
    int  res5;                   /* Reserved for 64 bit     */
    unsigned int Targ_Buf_ALET;   /* ALET of target buffer   */
    int  res6;                   /* Reserved for 64 bit     */
    unsigned long Targ_Buf_Len;   /* Length of target buffer */
    int  res7;                   /* Reserved for 64 bit     */
    void *Wrk_Buf_Ptr;           /* Pointer to Work Buffer   */
    int  res8;                   /* Reserved for 64 bit     */
}
```

```

unsigned int Wrk_Buf_ALET;          /* ALET of Work buffer      */
int res9;                          /* Reserved for 64 bit      */
unsigned long Wrk_Buf_Len;          /* Length of Work buffer    */
unsigned int ccsid_src;             /* str type source          */
unsigned int ccsid_trt;             /* str type target          */
struct {
    int Bidi_Context : 1,          /* Bidi Context             */
                                     /* 0 = Context LTR          */
                                     /* 1 = Context RTL          */
        Bidi_ImpAlg : 1,          /* Bidi Implicit Alg        */
                                     /* 0 = Algor Basic          */
                                     /* 1 = Algor Implicit       */
                                     : 6;
    } Flag1;
char res10[3];
int Return_Code;                   /* Return code              */
int Reason_Code;                   /* Reason code              */
}CUNBBPRM;

```

64-bit mapping

```

typedef struct tag_CUN4BBPR{
    int version;                   /* Parameter Area Version  */
    int length;                    /* Parameter Area Length    */
    void *Src_Buf_Ptr;             /* Pointer to Source        */
    int res1;
    unsigned int Src_Buf_ALET;      /* ALET of source buffer    */
    unsigned long Src_Buf_Len;      /* Length of source data    */
    void *Targ_Buf_Ptr;            /* Pointer to Target        */
    int res2;
    unsigned int Targ_Buf_ALET;     /* ALET of target buffer    */
    unsigned long Targ_Buf_Len;     /* Length of target buffer  */
    void *Wrk_Buf_Ptr;             /* Pointer to Work Buffer    */
    int res3;
    unsigned int Wrk_Buf_ALET;      /* ALET of Work buffer      */
    unsigned long Wrk_Buf_Len;      /* Length of Work buffer    */
    unsigned int ccsid_src;         /* str type source          */
    unsigned int ccsid_trt;         /* str type target          */
    struct {
        int Bidi_Context : 1,      /* Bidi Context             */
                                     /* 0 = Context LTR          */
                                     /* 1 = Context RTL          */
            Bidi_ImpAlg : 1,      /* Bidi Implicit Alg        */
                                     /* 0 = Algor Basic          */
                                     /* 1 = Algor Implicit       */
                                     : 6;
        } Flag1;
    char res4[3];
    int Return_Code;               /* Return code              */
    int Reason_Code;               /* Reason code              */
}CUN4BBPR;

```

Using the HLASM interface

This topic describes the syntax in HLASM to call stub routines for Bidi **CUNLBIDI** (AMODE (31)) and **CUN4LBID** (AMODE (64)).

For AMODE (31)

```

-----1-----2-----3-----4-----5-----6-----7--
GETMAIN .....Obtain storage for parameter area
*in primary address space.

```

```

LR    R4,R1    Save parameter area address
USING CUNBBPRM,R4    Make parameter area addressable

```

Bidi transformation

```
XC    CUNBBPRM,CUNBBPRM Init PARAMETER AREA TO BINARY 0
LA    R15,CUNBBPRM_VER  Get Version
ST    R15,CUNBBPRM_VERSION Store to parameter area
LA    R15,CUNBBPRM_LEN  Initialize Length
ST    R15,CUNBBPRM_LENGTH Move to parameter area
LA    R15,CUNBBPRM_SRCCSID Initialize String Type Src
ST    R15,CUNBBPRM_CC SID_Src
LA    R15,CUNBBPRM_TRGCCSID Initialize String Type Trg
ST    R15,CUNBBPRM_CC SID_Trt
```

*Supply source buffer pointer,length and ALET.
*Supply work buffer pointer,length and ALET.
*Supply target buffer pointer,length and ALET.
*Fill all required fields of the parameter area.

CALL CUNLBIDI,((R4)) Call stub routine with CUNBBPRM
*address as argument.

CUNBBIDF DSECT=YES Provide Mappings (CUNBBPRM,return and
*reason codes,constants for version
*and length).

For AMODE (64)

-----1-----2-----3-----4-----5-----6-----7--
GETMAIN Obtain storage for parameter area
*in primary address space.

```
LR    R4,R1    Save parameter area address
USING CUN4BBPR,R4 Make parameter area addressable
XC    CUN4BBPR,CUN4BBPR Init PARAMETER AREA TO BINARY 0
LA    R15,CUN4BBPR_VER  Get Version
ST    R15,CUN4BBPR_VERSION Store to parameter area
LA    R15,CUN4BBPR_LEN  Initialize Length
ST    R15,CUN4BBPR_LENGTH Move to parameter area
LA    R15,CUN4BBPR_SRCCSID Initialize String Type Src
ST    R15,CUN4BBPR_CC SID_Src
LA    R15,CUN4BBPR_TRGCCSID Initialize String Type Trg
ST    R15,CUN4BBPR_CC SID_Trt
```

*Supply source buffer pointer,length and ALET.
*Supply work buffer pointer,length and ALET.
*Supply target buffer pointer,length and ALET.
*Fill all required fields of the parameter area.

CALL CUN4LBID,((R4)) Call stub routine with CUN4BBPR
*address as argument.

CUN4BPID DSECT=YES Provide Mappings (CUN4BBPR,return and
*reason codes,constants for version
*and length).

Mapping of parameters for AMODE (31)

The mapping of the parameter areas is supplied by the interface definition file CUNBBIDF. This file is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

Table 39. Mapping of parameters in HLASM for Bidi AMODE (31)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Description
0	(0)	STRUCTURE	100	DWORD	CUNBBPRM	Parameter Area

Table 39. Mapping of parameters in HLASM for Bidi AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Description
0	(0)	UNSIGNED	4		CUNBBPRM_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUNBBPRM_Length	Parameter area Length
8	(8)	CHARACTER	4		*	Reserved for 64 bit
12	(0C)	ADDRESS	4		CUNBBPRM_Src_Buf_Ptr	Source buffer pointer
16	(0A)	CHARACTER	4		*	Reserved for 64 bit
20	(14)	UNSIGNED	4		CUNBBPRM_Src_Buf_ALET	Source buffer ALET
24	(18)	CHARACTER	4		*	Reserved for 64 bit
28	(1C)	UNSIGNED	4		CUNBBPRM_Src_Buf_Len	Source buffer length
32	(20)	CHARACTER	4		*	Reserved for 64 bit
36	(24)	ADDRESS	4		CUNBBPRM_Targ_Buf_Ptr	Target buffer pointer
40	(28)	CHARACTER	4		*	Reserved for 64 bit
44	(2C)	UNSIGNED	4		CUNBBPRM_Targ_Buf_ALET	Target buffer ALET
48	(30)	CHARACTER	4		*	Reserved for 64 bit
52	(34)	UNSIGNED	4		CUNBBPRM_Targ_Buf_Len	Target buffer length
56	(38)	CHARACTER	4		*	Reserved for 64 bit
60	(3C)	ADDRESS	4		CUNBBPRM_Wrk_Buf_Ptr	Work buffer pointer
64	(40)	CHARACTER	4		*	Reserved for 64 bit
68	(44)	UNSIGNED	4		CUNBBPRM_Wrk_Buf_ALET	Work buffer ALET
72	(48)	CHARACTER	4		*	Reserved for 64 bit
76	(4C)	UNSIGNED	4		CUNBBPRM_Wrk_Buf_Len	Work buffer length
80	(50)	UNSIGNED	4		CUNBBPRM_CCSID_Src	CCSID Source
84	(54)	UNSIGNED	4		CUNBBPRM_CCSID_Trg	CCSID Target
88	(58)	BITSTRING	1		CUNBBPRM_Flag1	FLAG Byte 1 set by caller
		1...			CUNBBPRM_Bidi_Context	Bidi Context: 0=Context LTR 1=Context RTL
		.1...			CUNBBPRM_Bidi_ImpAlg	Bidi Implicit Alg: 0=Algor Basic 1=Algor Implicit
89	(59)	CHARACTER	3		*	Reserved
92	(5C)	CHARACTER	8	WORD	CUNBBPRM_RC_RS	Return/reason code
		UNSIGNED	4		CUNBBPRM_Return_Code	Return code
		UNSIGNED	4		CUNBBPRM_Reason_Code	Reason code
100	(64)	CHARACTER	0		CUNBBPRM_End	End of CUNBBPRM

Description of parameters in area CUNBBPRM

This topic describes the fields in the parameter area for the Bidi service:

CUNBBPRM_Version - set by caller - Required

Specifies the version of the parameter area for Bidi.

Bidi transformation

CUNBBPRM_Length - set by caller - Required

Specifies the length of the parameter area.

CUNBBPRM_Src_Buf_Ptr - set by caller, updated by service - Required

Specifies the beginning address of a string of text characters, with a length specified in the CUNBBPRM_Src_Buf_Len parameter.

CUNBBPRM_Src_Buf_ALET - set by caller

Specifies the ALET to be used to access the source buffer addressed by CUNBBPRM_Src_Buf_Ptr.

CUNBBPRM_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUNBBPRM_Src_Buf_Ptr, to be transformed.

CUNBBPRM_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage to be used to store the final string layout.

CUNBBPRM_Targ_Buf_ALET - set by caller

Specifies the ALET to be used to access the target buffer addressed by CUNBBPRM_Targ_Buf_Ptr.

CUNBBPRM_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUNBBPRM_Targ_Buf_Ptr. It should be at least the same size of the CUNBBPRM_Src_Buf_Len.

CUNBBPRM_Wrk_Buf_Ptr - set by caller, used by service for conversion purposes.

Specifies the beginning address of an area of storage that the conversion services can use to store intermediate results.

CUNBBPRM_Wrk_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffer addressed by CUNBBPRM_Wrk_Buf_Ptr.

CUNBBPRM_Wrk_Buf_Len - set by caller

Specifies the length in bytes of the work buffer addressed by CUNBBPRM_Wrk_Buf_Ptr. It should be at least the same size of the CUNBBPRM_Src_Buf_Len.

CUNBBPRM_CCSID_Src - set by caller

Specifies the CCSID of the source.

CUNBBPRM_CCSID_Trg - set by caller

Specifies the CCSID of the target.

CUNBBPRM_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUNBBPRM_Bidi_context
x1xx xxxx	CUNBBPRM_Bidi_impalg

CUNBBPRM_Bidi_context

Specifies the context of the text to be transformed.

- **0**: Indicates the context is Left to Right (LTR).
- **1**: Indicates the context is Right to Left (RTL).

CUNBBPRM_Bidi_impalg

Specifies the algorithm to be used.

- 0: Indicates the basic algorithm will be used.
- 1: Indicates the implicit algorithm will be used.

CUNBBPRM_Return_Code - set by service

Specifies the return code.

CUNBBPRM_Reason_Code - set by service

Specifies the reason code.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas is supplied by the interface definition file CUN4BBID. This file is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

Table 40. Mapping of parameters in HLASM for Bidi AMODE (64)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Description
0	(0)	STRUCTURE	100	DWORD	CUN4BBPR	Parameter Area
0	(0)	UNSIGNED	4		CUN4BBPR_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUN4BBPR_Length	Parameter area Length
8	(8)	ADDRESS	8		CUN4BBPR_Src_Buf_Ptr	Source buffer pointer
16	(10)	CHARACTER	4		*	Reserved for 64 bit
20	(14)	UNSIGNED	4		CUN4BBPR_Src_Buf_ALET	Source buffer ALET
24	(18)	UNSIGNED	8		CUN4BBPR_Src_Buf_Len	Source buffer length
32	(20)	ADDRESS	8		CUN4BBPR_Targ_Buf_Ptr	Target buffer pointer
40	(28)	CHARACTER	4		*	Reserved for 64 bit
44	(2C)	UNSIGNED	4		CUN4BBPR_Targ_Buf_ALET	Target buffer ALET
48	(30)	UNSIGNED	8		CUN4BBPR_Targ_Buf_Len	Target buffer length
56	(38)	ADDRESS	8		CUN4BBPR_Wrk_Buf_Ptr	Work buffer pointer
64	(40)	CHARACTER	4		*	Reserved for 64 bit
68	(44)	UNSIGNED	4		CUN4BBPR_Wrk_Buf_ALET	Work buffer ALET
72	(48)	UNSIGNED	8		CUN4BBPR_Wrk_Buf_Len	Work buffer length
80	(50)	UNSIGNED	4		CUN4BBPR_CCSID_Src	CCSID Source
84	(54)	UNSIGNED	4		CUN4BBPR_CCSID_Trg	CCSID Target
88	(58)	BITSTRING	1		CUN4BBPR_Flag1	FLAG Byte 1 set by caller
		1... ..			CUN4BBPR_Bidi_Context	Bidi Context: 0=Context LTR 1=Context RTL
		.1... ..			CUN4BBPR_Bidi_ImpAlg	Bidi Implicit Alg: 0=Algor Basic 1=Algor Implicit
89	(59)	CHARACTER	3		*	Reserved
92	(5C)	CHARACTER	8	WORD	CUN4BBPR_RC_RS	Return/reason code
		UNSIGNED	4		CUN4BBPR_Return_Code	Return code
		UNSIGNED	4		CUN4BBPR_Reason_Code	Reason code
100	(64)	CHARACTER	0		CUN4BBPR_End	End of CUN4BBPR

Description of parameters in area CUN4BBPR

This topic describes the fields in the parameter area for the Bidi service:

CUN4BBPR_Version - set by caller - Required

Specifies the version of the parameter area for Bidi.

CUN4BBPR_Length - set by caller - Required

Specifies the length of the parameter area.

CUN4BBPR_Src_Buf_Ptr - set by caller, updated by service - Required

Specifies the beginning address of a string of text characters, with a length specified in the CUN4BBPR_Src_Buf_Len parameter.

CUN4BBPR_Src_Buf_ALET - set by caller

Specifies the ALET to be used to access the source buffer addressed by CUN4BBPR_Src_Buf_Ptr.

CUN4BBPR_Src_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the string in the source buffer, addressed by CUN4BBPR_Src_Buf_Ptr, to be transformed.

CUN4BBPR_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage to be used to store the final string layout.

CUN4BBPR_Targ_Buf_ALET - set by caller

Specifies the ALET to be used to access the target buffer addressed by CUN4BBPR_Targ_Buf_Ptr.

CUN4BBPR_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUN4BBPR_Targ_Buf_Ptr. It should be at least the same size of the CUN4BBPR_Src_Buf_Len.

CUN4BBPR_Wrk_Buf_Ptr - set by caller, used by service for conversion purposes.

Specifies the beginning address of an area of storage that the conversion services can use to store intermediate results.

CUN4BBPR_Wrk_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffer addressed by CUN4BBPR_Wrk_Buf_Ptr.

CUN4BBPR_Wrk_Buf_Len - set by caller

Specifies the length in bytes of the work buffer addressed by CUN4BBPR_Wrk_Buf_Ptr. It should be at least the same size of the CUN4BBPR_Src_Buf_Len.

CUN4BBPR_CCSID_Src - set by caller

Specifies the CCSID of the source.

CUN4BBPR_CCSID_Trg - set by caller

Specifies the CCSID of the target.

CUN4BBPR_Flag1 - set by caller

Bit position	Name
1xxx xxxx	CUN4BBPR_Bidi_context
x1xx xxxx	CUN4BBPR_Bidi_impalg

CUN4BBPR_Bidi_context

Specifies the context of the text to be transformed.

- **0**: Indicates the context is Left to Right (LTR).
- **1**: Indicates the context is Right to Left (RTL).

CUN4BBPR_Bidi_impalg

Specifies the algorithm to be used.

- **0**: Indicates the basic algorithm will be used.
- **1**: Indicates the implicit algorithm will be used.

CUN4BBPR_Return_Code - set by service

Specifies the return code.

CUN4BBPR_Reason_Code - set by service

Specifies the reason code.

Character conversion service and the new B technique

As mentioned in Character Conversion Service, Bidi Transformation Service can be called through CUNLCNV or CUN4LCNV by a special technique "B" that can be used along with the rest of the technique search order. For more information, see "Character conversion service and the new B technique" on page 57.

This new "B" technique is searched at the end of current "RECLM" search order when a technique search order has not been specified. Instead, it is used with RECLM. Bidi transformation services are called only when "B" is specified. Character conversion services work the same as specifying any of the existing techniques without technique "B".

End of Programming Interface information

Chapter 9. Stringprep conversion

Programming Interface information

This chapter describes the programming required for the stringprep conversion services.

Unicode System Services for International String preparation is also referred to as 'stringprep'. The stringprep conversion service can be called using a stub routine named CUNLSTRP for AMODE (31), and CUN4LSTP for AMODE (64).

The stringprep conversion service prepares a string of Unicode text in order to increase the likelihood that string input and string comparison work in ways that make sense for typical users. Preparation of Internationalized Strings better known as "Stringprep", is a way of preparing Unicode text strings in order to increase the likelihood that string input and string comparison work in ways that make sense for typical users throughout the world. The stringprep protocol is useful for protocol identifier values, company and personal names, internationalized domain names, and other text strings.

This z/OS Unicode implementation meets the specifications described in the RFC 3454. For further information about the string preparation standard, see <http://ietfreport.isoc.org/idref/rfc3454/>.

IMPORTANT: z/OS stringprep service requires the normalization services to be active on the current Unicode Environment. Also, ensure that Bidi services are installed on the system.

Calling the stringprep services

This is a general description of how the stringprep services are called.

The 31-bit caller has to provide:

- Profile Name (8 char string)
- Source buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Target buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Work1 buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Work2 buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- DDA buffer pointer (31-bit pointer), ALET (4 byte), and length (4 byte)
- Flags

The 64-bit caller has to provide:

- Profile Name (8 char string)
- Source buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Target buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Work1 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Work2 buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- DDA buffer pointer (64-bit pointer), ALET (4 byte), and length (8 byte)
- Flags

Using the C interface

This is the call syntax in C for calling the stub routine **CUNLSTRP** or **CUN4LSTP** (stringprep conversion). The mapping of the parameter area supplied by the header file `cunhc.h` is listed in “Mapping of parameters in C.”

```
#include<cunhc.h>
#define SLEN 1024
#define W1LEN 4096
#define W2LEN 4096
#define DDAL 4096
#define TLEN 4096
.....
unsigned char Sourcebuffer [SLEN];
unsigned char Workbuffer1 [W1LEN];
unsigned char Workbuffer2 [W2LEN];
unsigned char DDABuffer [DDAL];
unsigned char Targetbuffer [TLEN];

#ifdef _LP64 /* 64 bit */
CUN4BPPR myparm ={CUN4BPPR_DEFAULT};
#else /* 31 bit */
CUNBPPRM myparm ={CUNBPPRM_DEFAULT};
#endif

strcpy(Myparm.Profile_name,"CUNSTCIS");
Myparm.Src_Buf_Ptr = Sourcebuffer;
myparm.Wrk1_Buf_Ptr = Workbuffer1;
myparm.Wrk2_Buf_Ptr = Workbuffer2;
myparm.Targ_Buf_Ptr = Targetbuffer;
Myparm.Src_Buf_Len = SLEN;
Myparm.Wrk1_Buf_Len = W1LEN;
Myparm.Wrk2_Buf_Len = W2LEN;
myparm.Targ_Buf_Len = TLEN;

#ifdef _LP64 /* 31 bit */
CUNLSTRP(&myparm);
#else /* 64 bit */
CUN4LSTP(&myparm);
#endif

if((myparm.Return_Code != CUN_RC_OK).....
```

Mapping of parameters in C

A C header file is supplied (`cunhc.h`) that contains the function prototypes for the stringprep service. The following structure is used in the interface to the stringprep service.

31-bit mapping

```
typedef struct tag_CUNBPPRM{
    int version; /* Parameter Area Version */
    int length; /* Parameter Area Length */
    char prof_name[8]; /* Profile name */
    int res1; /* Reserved for 64 bit */
    void *Src_Buf_Ptr; /* Pointer to Source */
    int res2; /* Reserved for 64 bit */
    unsigned int Src_Buf_ALET; /* ALET of source buffer */
    int res3; /* Reserved for 64 bit */
    unsigned long Src_Buf_Len; /* Length of source data */
    int res4; /* Reserved for 64 bit */
    void *Targ_Buf_Ptr; /* Pointer to Target */
    int res5; /* Reserved for 64 bit */
}
```



```

unsigned int Targ_Buf_ALET; /* ALET of target buffer */
int res6; /* Reserved for 64 bit */
unsigned long Targ_Buf_Len; /* Length of target buffer */
int res7; /* Reserved for 64 bit */
void *Wrk1_Buf_Ptr; /* Pointer to Work1 Buffer */
int res8; /* Reserved for 64 bit */
unsigned int Wrk1_Buf_ALET; /* ALET of Work1 buffer */
int res9; /* Reserved for 64 bit */
unsigned long Wrk1_Buf_Len; /* Length of Work1 buffer */
int res10; /* Reserved for 64 bit */
void *Wrk2_Buf_Ptr; /* Pointer to Work2 Buffer */
int res11; /* Reserved for 64 bit */
unsigned int Wrk2_Buf_ALET; /* ALET of Work2 buffer */
int res12; /* Reserved for 64 bit */
unsigned long Wrk2_Buf_Len; /* Length of Work2 buffer */
int res13; /* Reserved for 64 bit */
void *DDA_Buf_Ptr; /* Pointer to DDA Buffer */
int res14; /* Reserved for 64 bit */
unsigned int DDA_Buf_ALET; /* ALET of DDA buffer */
int res15; /* Reserved for 64 bit */
unsigned long DDA_Buf_Len; /* Length of DDA buffer */
struct {
    UTF_version : 4, /* UTF version to use */
    /* 0 = UTF-8 */
    /* 1 = UTF-16 */
    UnassignedEr : 1, /* If an unassigned code */
    /* point found: */
    /* 0 = Terminate processing */
    /* and sets RC=8 */
    /* 1 = Continues processing */
    /* and sets RC=4 */
    Page_fix : 1, /* for Page fixing */
    /* 0 = No Page Fix */
    /* 1 = Page fix */
    : 2; /* FLAG Byte 1 set by caller */
} Flags; /* Flags */
unsigned char Res16[7]; /* Reserved */
int Return_Code; /* Return code */
int Reason_Code; /* Reason code */
}CUNBPPRM;

```

64-bit mapping

```

typedef struct tag_CUN4BPPR{
    int version; /* Parameter Area Version */
    int length; /* Parameter Area Length */
    char prof_name[8]; /* Profile name */
    void *Src_Buf_Ptr; /* Pointer to Source */
    int res1;
    unsigned int Src_Buf_ALET; /* ALET of source buffer */
    unsigned long Src_Buf_Len; /* Length of source data */
    void *Targ_Buf_Ptr; /* Pointer to Target */
    int res2;
    unsigned int Targ_Buf_ALET; /* ALET of target buffer */
    unsigned long Targ_Buf_Len; /* Length of target buffer */
    void *Wrk1_Buf_Ptr; /* Pointer to Work1 Buffer */
    int res3;
    unsigned int Wrk1_Buf_ALET; /* ALET of Work1 buffer */
    unsigned long Wrk1_Buf_Len; /* Length of Work1 buffer */
    void *Wrk2_Buf_Ptr; /* Pointer to Work2 Buffer */
    int res4;
    unsigned int Wrk2_Buf_ALET; /* ALET of Work2 buffer */
    unsigned long Wrk2_Buf_Len; /* Length of Work2 buffer */
    void *DDA_Buf_Ptr; /* Pointer to DDA Buffer */
    int res5;
    unsigned int DDA_Buf_ALET; /* ALET of DDA buffer */
}

```

Stringprep conversion

```
unsigned long DDA_Buf_Len;      /* Length of DDA buffer */
struct {
    UTF_version      : 4,      /* UTF version to use */
                                /* 0 = UTF-8 */
                                /* 1 = UTF-16 */
    UnassignedEr      : 1,      /* If an unassigned code */
                                /* point found: */
                                /* 0 = Terminate processing */
                                /* and sets RC=8 */
                                /* 1 = Continues processing */
                                /* and sets RC=4 */
    Page_fix          : 1,      /* for Page fixing */
                                /* 0 = No Page Fix */
                                /* 1 = Page fix */
                                : 2; /* FLAG Byte 1 set by caller*/
} Flags;                      /* Flags */
unsigned char Res6[7];         /* Reserved */
int Return_Code;              /* Return code */
int Reason_Code;              /* Reason code */
}CUN4BPPR;
```

Note: C constants for the parameter area are defined in the header file cunhc.h.

Using the HLASM interface

This topic describes the syntax in HLASM to call stub routines for stringprep **CUNLSTRP** (AMODE (31)), and **CUN4LSTP** (AMODE (64)).

For AMODE (31)

```
-----1-----2-----3-----4-----5-----6-----7--
GETMAIN .....Obtain storage for parameter area
*in primary address space.

LR   R4,R1    Save parameter area address
USING CUNBPPRM,R4 Make parameter area addressable
XC   CUNBBPRM(CUNBBPRM_LEN),CUNBBPRM Init PARAMETER AREA TO BINARY 0
LA   R15,CUNBPPRM_VER Get Version
ST   R15,CUNBPPRM_VERSION Store to parameter area
LA   R15,CUNBPPRM_LEN Initialize Length
ST   R15,CUNBPPRM_LENGTH Move to parameter area
MVC  CUNBPPRM_PROF_NAME,=CL8'CUNSTCIS' Provide profile name

*Supply source buffer pointer,length and ALET.
*Supply work buffer pointer,length and ALET.
*Supply target buffer pointer,length and ALET.
*Fill all required fields of the parameter area.

CALL CUNLSTRP,((R4)) Call stub routine with CUNBPPRM
*address as argument.

CUNBPIDF DSECT=YES Provide Mappings (CUNBPPRM,return and
*reason codes,constants for version
*and length).
```

For AMODE (64)

```
-----1-----2-----3-----4-----5-----6-----7--
GETMAIN .....Obtain storage for parameter area
*in primary address space.

LR   R4,R1    Save parameter area address
USING CUN4BPPR,R4 Make parameter area addressable
XC   CUN4BBPR(CUN4BBPR_LEN),CUN4BBPR CLEAR PARAMETER AREA
LA   R15,CUN4BPPR_VER Get Version
```

```

ST    R15,CUN4BPPR_VERSION Store to parameter area
LA    R15,CUN4BPPR_LEN  Initialize Length
ST    R15,CUN4BPPR_LENGTH Move to parameter area
MVC   CUN4BPPR_PROF_NAME,=CL8'CUNSTCIS' Provide profile name

```

```

*Supply source buffer pointer,length and ALET.
*Supply work buffer pointer,length and ALET.
*Supply target buffer pointer,length and ALET.
*Fill all required fields of the parameter area.

```

```

CALL CUN4LSTP,((R4)) Call stub routine with CUNBPPRM
*address as argument.

```

```

CUN4BPID DSECT=YES Provide Mappings (CUN4BPPR,return and
*reason codes,constants for version
*and length).

```

Mapping of parameters for AMODE (31)

The mapping of the parameter areas is supplied by the interface definition file CUNBPIDF. This file is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

Table 41. Mapping of parameters in HLASM for stringprep AMODE (31)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	156	DWORD	CUNBPPRM	Parameter Area
0	(0)	UNSIGNED	4		CUNBPPRM_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUNBPPRM_Length	Parameter area Length
8	(8)	CHARACTER	8		CUNBPPRM_Prof_Name	Profile name
16	(10)	CHARACTER	4		*	Reserved for 64 bit
20	(14)	ADDRESS	4		CUNBPPRM_Src_Buf_Ptr	Source buffer pointer
24	(18)	CHARACTER	4		*	Reserved for 64 bit
28	(1C)	UNSIGNED	4		CUNBPPRM_Src_Buf_ALET	Source buffer ALET
32	(20)	CHARACTER	4		*	Reserved for 64 bit
36	(24)	UNSIGNED	4		CUNBPPRM_Src_Buf_Len	Source buffer length
40	(28)	CHARACTER	4		*	Reserved for 64 bit
44	(2C)	ADDRESS	4		CUNBPPRM_Targ_Buf_Ptr	Target buffer pointer
48	(30)	CHARACTER	4		*	Reserved for 64 bit
52	(34)	UNSIGNED	4		CUNBPPRM_Targ_Buf_ALET	Target buffer ALET
56	(38)	CHARACTER	4		*	Reserved for 64 bit
60	(3C)	UNSIGNED	4		CUNBPPRM_Targ_Buf_Len	Target buffer length
64	(40)	CHARACTER	4		*	Reserved for 64 bit
68	(44)	ADDRESS	4		CUNBPPRM_Wrk1_Buf_Ptr	Wrk1 buffer pointer
72	(48)	CHARACTER	4		*	Reserved for 64 bit
76	(4C)	UNSIGNED	4		CUNBPPRM_Wrk1_Buf_ALET	Wrk1 buffer ALET
80	(50)	CHARACTER	4		*	Reserved for 64 bit
84	(54)	UNSIGNED	4		CUNBPPRM_Wrk1_Buf_Len	Wrk1 buffer length
88	(58)	CHARACTER	4		*	Reserved for 64 bit

Stringprep conversion

Table 41. Mapping of parameters in HLASM for stringprep AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
92	(5C)	ADDRESS	4		CUNBPPRM_Wrk2_Buf_Ptr	Wrk2 buffer pointer
96	(60)	CHARACTER	4		*	Reserved for 64 bit
100	(64)	UNSIGNED	4		CUNBPPRM_Wrk2_Buf_ALET	Wrk2 buffer ALET
104	(68)	CHARACTER	4		*	Reserved for 64 bit
108	(6C)	UNSIGNED	4		CUNBPPRM_Wrk2_Buf_Len	Wrk2 buffer length
112	(70)	CHARACTER	4		*	Reserved for 64 bit
116	(74)	ADDRESS	4		CUNBPPRM_DDA_Buf_Ptr	Dynamic data area pointer
120	(78)	CHARACTER	4		*	Reserved for 64 bit
124	(7C)	UNSIGNED	4		CUNBPPRM_DDA_Buf_ALET	Dynamic data area ALET
128	(80)	CHARACTER	4		*	Reserved for 64 bit
132	(84)	UNSIGNED	4		CUNBPPRM_DDA_Buf_Len	Dynamic data area length
136	(88)	CHARACTER	4		*	Reserved for 64 bit
140	(8C)	BITSTRING	1		CUNBPPRM_Flags	Flags
		000.			*	Reserved
		...1			CUNBPPRM_UTF_Version	UTF version to use: 0000 = UTF-8 0001 = UTF-16
		 1...			CUNBPPRM_UnassignedEr	If an unassigned code point found: 0 = Terminate processing and sets RC=8 1 = Continues processing
		1..			CUNBPPRM_Page_fix	Page fix: 0 = No Page fix 1 = Page fix
		11			*	Reserved
141	(8D)	CHARACTER	7		*	Reserved for 64 bit
148	(94)	CHARACTER	8	WORD	CUNBPPRM_RC_RS	Return/reason code
		UNSIGNED	4		CUNBPPRM_Return_Code	Return code
		UNSIGNED	4		CUNBPPRM_Reason_Code	Reason code
156	(9C)	CHARACTER	0		CUNBPPRM_End	End of CUNBPPRM

Description of parameters in area CUNBPPRM

This description applies to C and HLASM.

CUNBPPRM_Version - set by caller - Required

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUNLSTRP using the constant CUNBPPRM_Ver, which is supplied by the interface definition file CUNBPIDF.

CUNBPPRM_Length - set by caller - Required

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUNLSTRP using the constant CUNBPPRM_Len, which is supplied by the interface definition file CUNBPIDF.

CUNBPPRM_Prof_Name - set by caller - Required

Specifies the name of the profile to be applied on the Source buffer.

CUNBPPRM_Src_Buf_Ptr - set by caller, updated by service - Required

Specifies the beginning address of a string of text characters. At the completion of the stringprep, the service updates CUNBPPRM_Src_Buf_Ptr to point just past the last character that is successfully prepared.

CUNBPPRM_Src_Buf_ALET - set by caller

Specifies the ALET to be used to access the source buffer addressed by CUNBPPRM_Src_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUNBPPRM_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUNBPPRM_Src_Buf_Ptr, to be prepared.

CUNBPPRM_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage where the string text to be prepared is stored. At the completion of the preparation, the service updates CUNBPPRM_Targ_Buf_Ptr to point just past the last stored character, and updates CUNBPPRM_Targ_Buf_Len to indicate the number of bytes not yet consumed in the buffer.

CUNBPPRM_Targ_Buf_ALET - set by caller

Specifies the ALET to be used to access the target buffer addressed by CUNBPPRM_Targ_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUNBPPRM_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUNBPPRM_Targ_Buf_Ptr. It is strongly suggested this length be at least 4 times the size as CUNBPPRM_Src_Buf_Len.

CUNBPPRM_Wrk1_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the area of storage that the stringprep service can use to store intermediate results.

CUNBPPRM_Wrk1_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffers addressed by CUNBPPRM_Wrk1_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUNBPPRM_Wrk1_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work buffers addressed by CUNBPPRM_Wrk1_Buf_Ptr. It is strongly suggested this length to be the same size as CUNBPPRM_Targ_Buf_Len.

CUNBPPRM_Wrk2_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the area of storage that the stringprep service can use to store immediate results.

CUNBPPRM_Wrk2_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffers addressed by CUNBPPRM_Wrk2_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

Stringprep conversion

CUNBPPRM_Wrk2_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work buffers addressed by CUNBPPRM_Wrk2_Buf_Ptr. It is strongly suggested this length to be the same size as CUNBPPRM_Targ_Buf_Len.

CUNBPPRM_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the stringprep conversion service is using internally as dynamic data area.

Note: CUNBPPRM_DDA_Buf_Ptr must be double-word boundary.

CUNBPPRM_DDA_Buf_ALET - set by caller

Specifies the ALET to be used if the dynamic data area addressed by CUNBPPRM_DDA_Buf_Ptr resides in a different address or data space.

CUNBPPRM_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUNBPPRM_DDA_Buf_Ptr.

CUNBPPRM_Flags - set by caller

Bit position	Name
000x xxxx	Reserved
xxx1 xxxx	CUNBPPRM_UTF_Version
xxxx 1xxx	CUNBPPRM_UnAssignedEr
xxxx x1xx	CUNBPPRM_Page_Fix

Reserved

These flag bits are reserved for internal service use and should be set to 0.

CUNBPPRM_UTF_Version

Specifies UTF version source buffer is being passed to the service.

- **0:** UTF-8.
- **1:** UTF-16.

CUNBPPRM_UnAssignedEr

According to RFC 3454.

- **0:** Indicates that the string prep is to be terminated with an error.
- **1:** Indicates that the string prep is to be given a warning and continues processing.

CUNBPPRM_Page_Fix

Page fix information.

- **0:** Indicates the profile will not be stored on page fix.
- **1:** Indicates the profile will be stored on page fix.

Note: CUNBPPRM_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUNBPPRM_Return_Code - set by service

Specifies the return code.

CUNBPPRM_Reason_Code - set by service

Specifies the reason code.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas is supplied by the interface definition file CUN4BPID. This file is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

Table 42. Mapping of parameters in HLASM for stringprep AMODE (64)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	152	DWORD	CUN4BPPR	Parameter Area
0	(0)	UNSIGNED	4		CUN4BPPR_Version	Parameter Area VERSION
4	(4)	UNSIGNED	4		CUN4BPPR_Length	Parameter area Length
8	(8)	CHARACTER	8		CUN4BPPR_Prof_Name	Profile name
16	(10)	ADDRESS	8		CUN4BPPR_Src_Buf_Ptr	Source buffer pointer
24	(18)	CHARACTER	4		*	Reserved for 64 bit
28	(1C)	UNSIGNED	4		CUN4BPPR_Src_Buf_ALET	Source buffer ALET
32	(20)	UNSIGNED	8		CUN4BPPR_Src_Buf_Len	Source buffer length
40	(28)	ADDRESS	8		CUN4BPPR_Targ_Buf_Ptr	Target buffer pointer
48	(30)	CHARACTER	4		*	Reserved for 64 bit
52	(34)	UNSIGNED	4		CUN4BPPR_Targ_Buf_ALET	Target buffer ALET
56	(38)	UNSIGNED	8		CUN4BPPR_Targ_Buf_Len	Target buffer length
64	(40)	ADDRESS	8		CUN4BPPR_Wrk1_Buf_Ptr	Wrk1 buffer pointer
72	(48)	CHARACTER	4		*	Reserved for 64 bit
76	(4C)	UNSIGNED	4		CUN4BPPR_Wrk1_Buf_ALET	Wrk1 buffer ALET
80	(50)	UNSIGNED	8		CUN4BPPR_Wrk1_Buf_Len	Wrk1 buffer length
88	(58)	ADDRESS	8		CUN4BPPR_Wrk2_Buf_Ptr	Wrk2 buffer pointer
96	(60)	CHARACTER	4		*	Reserved for 64 bit
100	(64)	UNSIGNED	4		CUN4BPPR_Wrk2_Buf_ALET	Wrk2 buffer ALET
104	(68)	UNSIGNED	8		CUN4BPPR_Wrk2_Buf_Len	Wrk2 buffer length
112	(70)	ADDRESS	8		CUN4BPPR_DDA_Buf_Ptr	Dynamic data area pointer
120	(78)	CHARACTER	4		*	Reserved for 64 bit
124	(7C)	UNSIGNED	4		CUN4BPPR_DDA_Buf_ALET	Dynamic data area ALET
128	(80)	UNSIGNED	8		CUN4BPPR_DDA_Buf_Len	Dynamic data area length

Stringprep conversion

Table 42. Mapping of parameters in HLASM for stringprep AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
136	(88)	BITSTRING	1		CUN4BPPR_Flags	Flags
		000.			*	Reserved
		...1			CUN4BPPR_UTF_Version	UTF version to use: 0000 = UTF-8 0001 = UTF-16
		 1...			CUN4BPPR_UnassignedEr	If an unassigned code point found: 0 = Terminate processing and sets RC=8 1 = Continues processing
		1..			CUN4BPPR_Page_fix	Page fix: 0 = No Page fix 1 = Page fix
		11			*	Reserved
137	(89)	CHARACTER	7		*	Reserved for 64 bit
144	(90)	CHARACTER	8	WORD	CUN4BPPR_RC_RS	Return/reason code
		UNSIGNED	4		CUN4BPPR_Return_Code	Return code
		UNSIGNED	4		CUN4BPPR_Reason_Code	Reason code
152	(98)	CHARACTER	0		CUN4BPPR_End	End of CUN4BPPR

Description of parameters in area CUN4BPPR

This description applies to C and HLASM.

CUN4BPPR_Version - set by caller - Required

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUN4LSTP using the constant CUN4BPPR_Ver, which is supplied by the interface definition file CUN4BPID.

CUN4BPPR_Length - set by caller - Required

Specifies the length of the parameter area. HLASM users must initialize this field for the first call to CUN4LSTP using the constant CUN4BPPR_Len, which is supplied by the interface definition file CUN4BPID.

CUN4BPPR_Prof_Name - set by caller - Required

Specifies the name of the profile to be applied on the Source buffer.

CUN4BPPR_Src_Buf_Ptr - set by caller, updated by service - Required

Specifies the beginning address of a string of text characters. At the completion of the stringprep, the service updates CUN4BPPR_Src_Buf_Ptr to point just past the last character that is successfully prepared.

CUN4BPPR_Src_Buf_ALET - set by caller

Specifies the ALET to be used to access the source buffer addressed by CUN4BPPR_Src_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BPPR_Src_Buf_Len - set by caller

Specifies the length in bytes of the string in the source buffer, addressed by CUN4BPPR_Src_Buf_Ptr, to be prepared.

CUN4BPPR_Targ_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage where the string text to be prepared is stored. At the completion of the preparation, the service updates CUN4BPPR_Targ_Buf_Ptr to point just past the last stored character, and updates CUN4BPPR_Targ_Buf_Len to indicate the number of bytes not yet consumed in the buffer.

CUN4BPPR_Targ_Buf_ALET - set by caller

Specifies the ALET to be used to access the target buffer addressed by CUN4BPPR_Targ_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BPPR_Targ_Buf_Len - set by caller

Specifies the length in bytes of the target buffer addressed by CUN4BPPR_Targ_Buf_Ptr. It is strongly suggested this length be at least 4 times the size as CUN4BPPR_Src_Buf_Len.

CUN4BPPR_Wrk1_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the area of storage that the stringprep service can use to store intermediate results.

CUN4BPPR_Wrk1_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffers addressed by CUN4BPPR_Wrk1_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BPPR_Wrk1_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work buffers addressed by CUN4BPPR_Wrk1_Buf_Ptr. It is strongly suggested this length to be the same size as CUN4BPPR_Targ_Buf_Len.

CUN4BPPR_Wrk2_Buf_Ptr - set by caller, updated by service

Specifies the beginning address of the area of storage that the stringprep service can use to store immediate results.

CUN4BPPR_Wrk2_Buf_ALET - set by caller

Specifies the ALET to be used to access the work buffers addressed by CUN4BPPR_Wrk2_Buf_Ptr. Use an ALET value of 0 to designate the primary address space.

CUN4BPPR_Wrk2_Buf_Len - set by caller, updated by service

Specifies the length in bytes of the work buffers addressed by CUN4BPPR_Wrk2_Buf_Ptr. It is strongly suggested this length to be the same size as CUN4BPPR_Targ_Buf_Len.

CUN4BPPR_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the stringprep conversion service is using internally as dynamic data area.

Note: CUN4BPPR_DDA_Buf_Ptr must be double-word boundary.

CUN4BPPR_DDA_Buf_ALET - set by caller

Specifies the ALET to be used if the dynamic data area addressed by CUN4BPPR_DDA_Buf_Ptr resides in a different address or data space.

CUN4BPPR_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUN4BPPR_DDA_Buf_Ptr.

Stringprep conversion

CUN4BPPR_Flags - set by caller

Bit position	Name
000x xxxx	Reserved
xxx1 xxxx	CUN4BPPR_UTF_Version
xxxx 1xxx	CUN4BPPR_UnAssignedEr
xxxx x1xx	CUN4BPPR_Page_Fix

Reserved

These flag bits are reserved for internal service use and should be set to 0.

CUN4BPPR_UTF_Version

Specifies UTF version source buffer is being passed to the service.

- **0:** UTF-8.
- **1:** UTF-16.

CUN4BPPR_UnAssignedEr

According to RFC 3454.

- **0:** Indicates that the string prep is to be terminated with an error.
- **1:** Indicates that the string prep is to be given a warning and continues processing.

CUN4BPPR_Page_Fix

Page fix information.

- **0:** Indicates the profile will not be stored on page fix.
- **1:** Indicates the profile will be stored on page fix.

Note: CUN4BPPR_Page_Fix applies to callers that run from Key 0 to Key 7 only. Callers with other keys (8-F) cannot exploit PAGE FIX storage in the Unicode Data Space.

CUN4BPPR_Return_Code - set by service

Specifies the return code.

CUN4BPPR_Reason_Code - set by service

Specifies the reason code.

Sample programs

Sample programs for Stringprep services are provided in SYS1.SAMPLIB:

- CUNSPSMC for C
- CUNSPSMA for HLASM

_____ **End of Programming Interface information** _____

Chapter 10. Conversion information service

This chapter describes the programming required for the conversion information service.

You can use the conversion information service to obtain information about details of one specific coded character set identifier (CCSID) or two CCSIDs. Use the conversion information service separately, or use the service before the z/OS Unicode character conversion service. The conversion information services are called using a stub routine named **CUNLINFO** for AMODE (31) and **CUN4LINP** for AMODE (64). Callers for conversion information service must provide at least one CCSID to obtain the following CCSID information:

- Encoding scheme ID and encoding scheme name
- Encoding Minimum size and maximum size
- CCSID description
- Number of substitution characters and these substitution characters
- SubCCSIDs information (if any)
- Supported CCSID or unsupported CCSID

When two CCSIDs are provided, and these CCSIDs are supported, conversion information service returns the techniques supported between those CCSIDs in addition to the CCSID information for each one of them.

Calling the conversion information service

This is a general description of how to call the conversion information services.

The 31 bit caller has to provide the following information:

- Parameter area version.
- Dynamic data area pointer (31 bit pointer), ALET (4 byte), and length (4 byte).
- SubCCSID buffer pointer (31 bit pointer), ALET (4 byte) - This is optional.
- One or more CCSIDs to retrieve information.
- Flags. Specifies whether techniques supported can be retrieved from CCSID2 to CCSID1 and from CCSID1 to CCSID2.

The 64-bit caller has to provide the following information:

- Parameter area version.
- Dynamic data area pointer (64 bit pointer), ALET (4 byte), and length (4 byte).
- SubCCSID buffer pointer (64 bit pointer), ALET (4 byte). This is optional.
- One or more CCSIDs to retrieve information.
- Flags. Specifies whether techniques supported can be retrieved from CCSID2 to CCSID1 and from CCSID1 to CCSID2.

Restrictions for the calling environment

Table 43. Restrictions while calling the conversion information service services

Property	Restriction
Authorization	Problem state or supervisor state, and any PSW key
Dispatchable unit mode	Task or SRB

Table 43. Restrictions while calling the conversion information service services (continued)

Property	Restriction
Cross memory mode	Any PASN, any HASN, any SASN
AMODE	31-bit and 64-bit
ASC mode	Called in primary mode but using AR mode
Interrupt status	Enabled for I/O and external interrupts.
Locks	May be held by the caller, but is not required to hold any
Control parameters	Must be in the primary address space
Recovery environment	Provided exclusively by the caller of the conversion services

Using the C interface

This is the call syntax in C for calling the stub routine **CUNLINFO** (conversion information service). The mapping of the parameter area supplied by the header file `cunhc.h` (SYS1.SCUNHF) is listed in "Mapping of parameters in C." A sample program, `CUNSISMC`, is provided in `SYS1.SAMPLIB`.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#include "cunhc.h"
.....
CUNBIPRM MyCInfParm = {CUNBIPRM_DEFAULT};
char DDA[CUNBIPRM_DDA_REQ];
char subCCSIDsBuffer[CUNBIPRM_SUBCCSIDS_INFO_LEN_REQ];
CUNBIPRM_subCCSIDs_Info * subCCSIDsBuff;
MyCInfParm.DDA_Buf_Ptr = DDA;
MyCInfParm.DDA_Buf_Len = CUNBIPRM_DDA_REQ;
memset(DDA, '\x00', CUNBIPRM_DDA_REQ);
memset(subCCSIDsBuffer, '\x00', CUNBIPRM_SUBCCSIDS_INFO_LEN_REQ);
MyCInfParm.CCSID1_subCCSIDs_Info_Ptr = subCCSIDsBuffer;
MyCInfParm.CCSID1_subCCSIDs_Info_ALET = 0;
MyCInfParm.CCSID1 = 1047;
MyCInfParm.CCSID2 = 0;
CUNLINFO(&MyCInfParm);
if (MyCInfParm.Gen_Flags_Out.CCSID1_Supported).....
```

Mapping of parameters in C

A C header file `cunhc.h` is supplied that contains the function prototypes for the conversion information service. The following structure is used in the interface to the conversion information service.

31-bit mapping

```
typedef struct tagCUNBIPRM {
    unsigned int    Version;                /* Structure version number */
    unsigned int    Length;                 /* Length of structure */
    unsigned int    CCSID1;                 /* CCSID1 Info ----- */
    struct {
        char        Res1[2];               /* Reserved */
        short        int CCSID1_ES_ID;      /* Encoding Scheme ID */
        char        CCSID1_ES_Name[28];    /* Encoding Scheme Name */
    } CCSID1_ES;                          /* CCSID1 Encoding Scheme info */
}
```

```

struct {
    unsigned char CCSID1_ES_Size_Min; /* ES Size Min */
    unsigned char CCSID1_ES_Size_Max; /* ES Size Max */
} CCSID1_ES_Size; /* Encoding scheme size */
char Res2[2]; /* Reserved */
char CCSID1_Description[64]; /* CCSID1 Description */
struct {
    unsigned char CCSID1_Num_Subs_SBCS; /* Num of Subs for SBCS */
    unsigned char CCSID1_Num_Subs_DBCS; /* Num of Subs for DBCS */
    unsigned char CCSID1_Num_Subs_TBCS; /* Num of Subs for TBCS */
    unsigned char CCSID1_Num_Subs_QBCS; /* Num of Subs for QBCS */
} CCSID1_Num_Subs; /* Reserved */
/* Num of Subs per Code Set */
struct {
    struct {
        char CCSID1_Sub_Char_SBCS_1[1];
        char CCSID1_Sub_Char_SBCS_2[1];
    } CCSID1_Sub_Char_SBCS; /* SBCS subs chars - right aligned */

    struct {
        char CCSID1_Sub_Char_DBCS_1[2];
        char CCSID1_Sub_Char_DBCS_2[2];
    } CCSID1_Sub_Char_DBCS; /* DBCS subs chars - right aligned */

    struct {
        char CCSID1_Sub_Char_TBCS_1[3];
        char CCSID1_Sub_Char_TBCS_2[3];
    } CCSID1_Sub_Char_TBCS; /* TBCS subs chars - right aligned */

    struct {
        char CCSID1_Sub_Char_QBCS_1[4];
        char CCSID1_Sub_Char_QBCS_2[4];
    } CCSID1_Sub_Char_QBCS; /* QBCS subs chars - right aligned */
    char Res4[4]; /* Reserved */
} CCSID1_Sub_Char; /* Substitution characters per CS */
char Res5[4]; /* Reserved */
void * CCSID1_subCCSIDs_Info_Ptr; /* Pointer to CUNBIPRM_subCCSIDs_Info (Optional) */
unsigned int CCSID1_subCCSIDs_Info_ALET; /* ALET for CCSID1_subCCSIDs_Info_Ptr */
unsigned char CCSID1_subCCSIDs_Info_Num; /* Num of subCCSIDs */
char Res6[3]; /* Reserved */
/* CCSID2 Info ----- */
unsigned int CCSID2; /* CCSID2 */
struct {
    char Res1a[2]; /* Reserved */
    short int CCSID2_ES_ID; /* Encoding Scheme ID */
    char CCSID2_ES_Name[28]; /* Encoding Scheme Name */
} CCSID2_ES; /* CCSID2 Encoding Scheme info */

struct {
    unsigned char CCSID2_ES_Size_Min; /* ES Size Min */
    unsigned char CCSID2_ES_Size_Max; /* ES Size Max */
} CCSID2_ES_Size; /* Encoding scheme size */
char Res2a[2]; /* Reserved */
char CCSID2_Description[64]; /* CCSID2 Description */
struct {
    unsigned char CCSID2_Num_Subs_SBCS; /* Num of Subs for SBCS */
    unsigned char CCSID2_Num_Subs_DBCS; /* Num of Subs for DBCS */
    unsigned char CCSID2_Num_Subs_TBCS; /* Num of Subs for TBCS */
    unsigned char CCSID2_Num_Subs_QBCS; /* Num of Subs for QBCS */
} CCSID2_Num_Subs; /* Reserved */
/* Num of Subs per Code Set */
struct {
    struct {
        char CCSID2_Sub_Char_SBCS_1[1];
        char CCSID2_Sub_Char_SBCS_2[1];
    }

```

Conversion information service

```

    } CCSID2_Sub_Char_SBCS;                /* SBCS subs chars - right aligned */

    struct {
        char CCSID2_Sub_Char_DBCS_1[2];
        char CCSID2_Sub_Char_DBCS_2[2];
    } CCSID2_Sub_Char_DBCS;                /* DBCS subs chars - right aligned */

    struct {
        char CCSID2_Sub_Char_TBCS_1[3];
        char CCSID2_Sub_Char_TBCS_2[3];
    } CCSID2_Sub_Char_TBCS;                /* TBCS subs chars - right aligned */

    struct {
        char CCSID2_Sub_Char_QBCS_1[4];
        char CCSID2_Sub_Char_QBCS_2[4];
    } CCSID2_Sub_Char_QBCS;                /* QBCS subs chars - right aligned */
    char Res4a[4];                          /* Reserved */
    } CCSID2_Sub_Char;                      /* Substitution characters per CS */
char Res5a[4];                            /* Reserved */
void * CCSID2_subCCSIDs_Info_Ptr ;         /* Pointer to
/* CUNBIPRM_subCCSIDs_Info (Optional) */
unsigned int  CCSID2_subCCSIDs_Info_ALET;  /* ALET for
/* CCSID2_subCCSIDs_Info_Ptr */
unsigned char CCSID2_subCCSIDs_Info_Num ;  /* Num of subCCSIDs
char          Res6a[3];                  /* Reserved
/* Conversion Info ----- */

    struct {
        int CCSID1_Supported      : 1,    /* CCSID1 Supported:
/* 0 - CCSID1 not supported
/* 1 - CCSID1 supported
/* Note. Meaningful if CCSID1
/*      was provided only
/*
        CCSID2_Supported      : 1,        /* CCSID2 Supported:
/* 0 - CCSID2 not supported
/* 1 - CCSID2 supported
/* Note. Meaningful if CCSID2
/*      was provided only
/*
        Conversion_Supported : 1,         /* Conversion From CCSID TO
/* CCSID2 supported:
/* 0 = No
/* 1 = Yes
/* Note. Meaningful in case that
/*      CCSID1 and CCSID2 are
/*      provided
/*
        : 5;                             /* Reserved
/* Out Flags-Set by the Service
    } Gen_Flags_Out;
    struct {
        int Get_Tech_Supp_fCCSID2_tCCSID1
        : 1,
        : 7;
    } Gen_Flags_In;
char Res7[6];
char Conv_Tech_fCCSID1_tCCSID2[8];
/* Conversion techniques sup-
/* ported From CCSID1 To
/* CCSID2
/* Note. Meaningful in case that
/*      Conversion_Supported is
/*      Turned ON
char Conv_Tech_fCCSID2_tCCSID1[8];
/* Conversion techniques sup-
/* ported From CCSID2 To
/* CCSID1
/* Note. Meaningful in case that

```

```

/* Conversion_Supported is
/* Turned ON */

/* DDA Info ----- */

char Res8[4];
void * DDA_Buf_Ptr;
unsigned int DDA_Buf_ALET;
unsigned int DDA_Buf_Len;

/* Reserved */
/* Dynamic data area pointer */
/* Dynamic data area ALET */
/* Dynamic data area length */
/* RC / RS */

struct {
    int Return_Code;
    int Reason_Code;
} RC_RS;
/* Return_Code */
/* Reason_Code */
/* Return/Reason code */
} CUNBIPRM;

```

Note: C constants for the parameter area are defined in the header file cunhc.h.

64-bit mapping

```

typedef struct tagCUN4BIPR {
    unsigned int Version;
    unsigned int Length;

    unsigned int CCSID1;
    struct {
        char Res1[2];
        short int CCSID1_ES_ID;
        char CCSID1_ES_Name[28];
    } CCSID1_ES;

    struct {
        unsigned char CCSID1_ES_Size_Min;
        unsigned char CCSID1_ES_Size_Max;
    } CCSID1_ES_Size;
    char Res2[2];
    char CCSID1_Description[64];
    struct {
        unsigned char CCSID1_Num_Subs_SBCS;
        unsigned char CCSID1_Num_Subs_DBCS;
        unsigned char CCSID1_Num_Subs_TBCS;
        unsigned char CCSID1_Num_Subs_QBCS;
        char Res3[4];
    } CCSID1_Num_Subs;

    struct {
        struct {
            char CCSID1_Sub_Char_SBCS_1[1];
            char CCSID1_Sub_Char_SBCS_2[1];
        } CCSID1_Sub_Char_SBCS;

        struct {
            char CCSID1_Sub_Char_DBCS_1[2];
            char CCSID1_Sub_Char_DBCS_2[2];
        } CCSID1_Sub_Char_DBCS;

        struct {
            char CCSID1_Sub_Char_TBCS_1[3];
            char CCSID1_Sub_Char_TBCS_2[3];
        } CCSID1_Sub_Char_TBCS;

        struct {
            char CCSID1_Sub_Char_QBCS_1[4];
            char CCSID1_Sub_Char_QBCS_2[4];
        } CCSID1_Sub_Char_QBCS;
        char Res4[4];
    } CCSID1_Sub_Char;
} CUN4BIPR;

```

Conversion information service

```

void * CCSID1_subCCSIDs_Info_Ptr ;          /* Pointer to                               */
/* CUN4BIPR_subCCSIDs_Info (Optional) */
unsigned int CCSID1_subCCSIDs_Info_ALET; /* ALET for                               */
/* CCSID1_subCCSIDs_Info_Ptr           */
unsigned char CCSID1_subCCSIDs_Info_Num ; /* Num of subCCSIDs                       */
char Res5[3];                             /* Reserved                               */
/* CCSID2 Info ----- */
unsigned int CCSID2;                       /* CCSID2                                 */
struct {
    char Res1a[2];                         /* Reserved                               */
    short int CCSID2_ES_ID;                /* Encoding Scheme ID                     */
    char CCSID2_ES_Name[28];               /* Encoding Scheme Name                   */
} CCSID2_ES;                              /* CCSID2 Encoding Scheme info           */
struct {
    unsigned char CCSID2_ES_Size_Min;      /* ES Size Min                           */
    unsigned char CCSID2_ES_Size_Max;      /* ES Size Max                           */
} CCSID2_ES_Size;                         /* Encoding scheme size                   */
char Res2a[2];                             /* Reserved                               */
char CCSID2_Description[64];               /* CCSID2 Description                     */
struct {
    unsigned char CCSID2_Num_Subs_SBCS;    /* Num of Subs for SBCS                   */
    unsigned char CCSID2_Num_Subs_DBCS;    /* Num of Subs for DBCS                   */
    unsigned char CCSID2_Num_Subs_TBCS;    /* Num of Subs for TBCS                   */
    unsigned char CCSID2_Num_Subs_QBCS;    /* Num of Subs for QBCS                   */
    char Res3a[4];                         /* Reserved                               */
} CCSID2_Num_Subs;                        /* Num of Subs per Code Set              */
struct {
    struct {
        char CCSID2_Sub_Char_SBCS_1[1];
        char CCSID2_Sub_Char_SBCS_2[1];
    } CCSID2_Sub_Char_SBCS;                /* SBCS subs chars - right aligned */
    struct {
        char CCSID2_Sub_Char_DBCS_1[2];
        char CCSID2_Sub_Char_DBCS_2[2];
    } CCSID2_Sub_Char_DBCS;                /* DBCS subs chars - right aligned */
    struct {
        char CCSID2_Sub_Char_TBCS_1[3];
        char CCSID2_Sub_Char_TBCS_2[3];
    } CCSID2_Sub_Char_TBCS;                /* TBCS subs chars - right aligned */
    struct {
        char CCSID2_Sub_Char_QBCS_1[4];
        char CCSID2_Sub_Char_QBCS_2[4];
    } CCSID2_Sub_Char_QBCS;                /* QBCS subs chars - right aligned */
    char Res4a[4];                         /* Reserved                               */
} CCSID2_Sub_Char;                         /* Substitution characters per CS        */
void * CCSID2_subCCSIDs_Info_Ptr ;          /* Pointer to                               */
/* CUN4BIPR_subCCSIDs_Info (Optional) */
unsigned int CCSID2_subCCSIDs_Info_ALET; /* ALET for                               */
/* CCSID2_subCCSIDs_Info_Ptr           */
unsigned char CCSID2_subCCSIDs_Info_Num; /* Num of subCCSIDs                       */
char Res5a[3];                             /* Reserved                               */
/* Conversion Info ----- */
struct {
    int CCSID1_Supported : 1,              /* CCSID1 Supported:                     */
/* 0 - CCSID1 not supported */
/* 1 - CCSID1 supported */
/* Note. Meaningful if CCSID1 */
/* was provided only */
    CCSID2_Supported : 1,                 /* CCSID2 Supported:                     */
/* 0 - CCSID2 not supported */
/* 1 - CCSID2 supported */
/* Note. Meaningful if CCSID2 */
/* was provided only */
    Conversion_Supported : 1,             /* Conversion From CCSID TO              */

```



```

/* CCSID2 supported: */
/* 0 = No */
/* 1 = Yes */
/* Note. Meaningful in case that */
/* CCSID1 and CCSID2 are */
/* provided */
/* Reserved */
/* Out Flags-Set by the Service */

: 5;
} Gen_Flags_Out;
struct {
    int Get_Tech_Supp_fCCSID2_tCCSID1
        : 1,

        : 7;
} Gen_Flags_In;
char Res6[6];
char Conv_Tech_fCCSID1_tCCSID2[8];

char Conv_Tech_fCCSID2_tCCSID1[8];

void *      DDA_Buf_Ptr;
unsigned int DDA_Buf_ALET;
unsigned int DDA_Buf_Len;

struct {
    int Return_Code;
    int Reason_Code;
} RC_RS;
} CUN4BIPR;

```

Using the HLASM interface

This is the call syntax in HLASM for calling the stub routine **CUNLINFO** (conversion information service for 31-bit callers) and **CUN4LINF** (conversion information service for 64-bit callers). A sample program, CUNSI SMA, is provided in SYS1.SAMPLIB.

For AMODE (31)

```

-----1-----2-----3-----4-----5-----6-----7--
      EJECT
CUNSI SMA CSECT
CUNSI SMA AMODE 31
CUNSI SMA RMODE ANY
      SPACE 1
      BRAS R15,PSTART      ! ESTABLISH ADDRESSABILITY
PSTART EQU *
      USING PSTART,R15
      B START
SAVE DC 36F'0'
START DS 0H
      STM R14,R12,12(R13) ! STORE CALLERS REGS
      LA R10,SAVE

```

```

        USING SAVEAREA,R10      ! ESTABLISH ADDRESSABILITY
        SPACE 1
        ST  R13,PREVSA          ! CHAIN CALLER'S SAVEAREA ADDRESS
        ST  R10,NEXTSA          ! TO CURRENT SAVERAREA
        LR  R13,R10             ! LET R13 POINT TO CURRENT SAVEAREA
        DROP R15,R10
        SPACE 1
        LAE R12,0(R15,0)        ! LOAD BASE AND CLEAR ACCESS REGISTER
        USING PSTART,R12
        SPACE 1
*****
* PREPARE PARAMETER AREA FOR CALL TO THE CONVERSION ROUTINES *
*****
        SPACE 1
        LA  R8,PARMAREA          ! GET PARAMETER AREA ADDR
        USING CUNBIPRM,R8       ! ESTABLISH ADDRESSABILITY
        SPACE 1
*
        LAE R2,CUNBIPRM          ! CLEAR PARAMETER AREA
        LHI R3,CUNBIPRM_LEN      ! PA Address
        LHI R15,0                ! PA Len
        MVCL R2,R14              ! Filler - Nulls
        SPACE 1
*
        LA  R0,CUNBIPRM_VER      ! SETTING PA VERSION
        ST  R0,CUNBIPRM_VERSION  ! GET ACTUAL VERSION
        LA  R0,CUNBIPRM_LEN      ! STORE INTO PARAMETER
        ST  R0,CUNBIPRM_LENGTH  ! GET ACTUAL LENGTH
        ST  R0,CUNBIPRM_LENGTH  ! STORE INTO PARAMETER
*
*                               /*****
*                               /*   Setting CCSIDs   */
*                               *****/
*
        SPACE 2
        LA  R0,CCSID1            ! Loading CCSID1
        ST  R0,CUNBIPRM_CCSID1  ! Setting CCSID1
        SPACE 2
        LA  R0,CCSID2            ! Loading CCSID2
        ST  R0,CUNBIPRM_CCSID2  ! Setting CCSID2
*****
* IMPORTANT: A DDA IS ALWAYS REQUIRED *
*****
*                               /*****
*                               /*   Setting DDA buffers   */
*                               *****/
*
        SPACE 2
        SR  R0,R0
        L   R0,ADDA
        ST  R0,CUNBIPRM_DDA_BUF_PTR
        MVC CUNBIPRM_DDA_BUF_ALET,=F'0'
        L   R0,=A(CUNBIPRM_DDA_REQ)
        ST  R0,CUNBIPRM_DDA_BUF_LEN
        SPACE 1
*****
* CALLING THE CNV INFO SERVICE *
*****
        SPACE 1
        CALL CUNLINFO,PARMAREA
        SPACE 1
EXIT    DS  0H
        LM  R15,R0,CUNBIPRM_RC_RS ! SET RETURN AND REASON CODE
        L   R13,4(R13)            ! RESTORE CALLER'S R13
        L   R14,12(R13)          ! RESTORE R14
        LM  R1,R12,24(R13)       ! RESTORE R1-R12 (RETAIN
*                               ! R15 AND R0)
        BR  R14
        SPACE 1
        LTORG ,

```

```

SPACE 1
*****
*
*           D E C L A R A T I O N           *
*
*
*           S e c t i o n                     *
*
*
*****

*****
*           CONSTANT CUNBIPRM_LEN IS USED TO ENSURE THAT SUFFICIENT *
*           STORAGE IS OBTAINED FOR THE PARAMETER AREA.           *
*****

SPACE 1
PARMAREA DC (CUNBIPRM_LEN)X'00' ! STORAGE FOR PARAMETER AREA
ADDA     DC A(DDA)                ! ADDRESS OF DDA

*****
*           CONSTANT CUNBIPRM_DDA_REQ IS USED TO ENSURE THAT SUFFICIENT *
*           STORAGE FOR THE DDA IS OBTAINED.                             *
*****

SPACE 1
DDA      DC (CUNBIPRM_DDA_Req)X'00' ! DDA SIZE
CCSID1   DC F'1047'                ! CCSID1
CCSID2   DC F'1208'                ! CCSID2
SAVEAREA DSECT
DC F'0' ! RESERVED
PREVSA   DC F'0' ! ADDRESS OF PREVIOUS SAVEAREA
NEXTSA   DC F'0' ! ADDRESS OF NEXT SAVEAREA
SAVER14  DC F'0'
SAVER15  DC F'0'
SAVER0   DC F'0'
SAVER1   DC F'0'
SAVER2   DC F'0'
SAVER3   DC F'0'
SAVER4   DC F'0'
SAVER5   DC F'0'
SAVER6   DC F'0'
SAVER7   DC F'0'
SAVER8   DC F'0'
SAVER9   DC F'0'
SAVER10  DC F'0'
SAVER11  DC F'0'
SAVER12  DC F'0'

SPACE 1
COPY CUNBIIDF
SPACE 1
CUNBIIDF DSECT=YES,LIST=YES
SPACE 1
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12

```

Conversion information service

```
R13    EQU    13
R14    EQU    14
R15    EQU    15
END CUNSISMA
```

For AMODE (64)

```
-----1-----2-----3-----4-----5-----6-----7--
      EJECT
CUNSISMA CSECT
CUNSISMA AMODE 31
CUNSISMA RMODE ANY
      SPACE 1
      BRAS R15,PSTART      ! ESTABLISH ADDRESSABILITY
PSTART EQU    *
      USING PSTART,R15
      B      START
SAVE    DC    36F'0'
START   DS    0H
      STM    R14,R12,12(R13) ! STORE CALLERS REGS
      LA     R10,SAVE
      USING SAVEAREA,R10    ! ESTABLISH ADDRESSABILITY
      SPACE 1
      ST     R13,PREVSA     ! CHAIN CALLER'S SAVEAREA ADDRESS
      ST     R10,NEXTSA     ! TO CURRENT SAVERAREA
      LR     R13,R10        ! LET R13 POINT TO CURRENT SAVEAREA
      DROP   R15,R10
      SPACE 1
      LAE    R12,0(R15,0)    ! LOAD BASE AND CLEAR ACCESS REGISTER
      USING PSTART,R12
      SPACE 1
*****
* PREPARE PARAMETER AREA FOR CALL TO THE CONVERSION ROUTINES *
*****
      SPACE 1
      LA     R8,PARMAREA     ! GET PARAMETER AREA ADDR
      USING CUN4BIPR,R8     ! ESTABLISH ADDRESSABILITY
      SPACE 1
*
      LAE    R2,CUN4BIPR     ! PA Address
      LHI    R3,CUN4BIPR_LEN ! PA Len
      LHI    R15,0           ! Filler - Nulls
      MVCL   R2,R14          ! Cleaning...
      SPACE 1
*
      LA     R0,CUN4BIPR_VER ! GET ACTUAL VERSION
      ST     R0,CUN4BIPR_VERSION ! STORE INTO PARAMETER
      LA     R0,CUN4BIPR_LEN ! GET ACTUAL LENGTH
      ST     R0,CUN4BIPR_LENGTH ! STORE INTO PARAMETER
*
*                                     /*****
*                                     /*      Setting CCSIDs      */
*                                     *****/
      SPACE 2
      LA     R0,CCSID1       ! Loading CCSID1
      ST     R0,CUN4BIPR_CCSID1 ! Setting CCSID1
      SPACE 2
      LA     R0,CCSID2       ! Loading CCSID2
      ST     R0,CUN4BIPR_CCSID2 ! Setting CCSID2
*****
* IMPORTANT: A DDA IS ALWAYS REQUIRED *
*****
*                                     /*****
*                                     /*      Setting DDA buffers    */
*                                     *****/
      SPACE 2
      SR     R0,R0
```

```

L      R0,ADDA
ST     R0,CUN4BIPR_DDA_BUF_PTR
MVC    CUN4BIPR_DDA_BUF_ALÉT,=F'0'
L      R0,=A(CUN4BIPR_DDA_REQ)
ST     R0,CUN4BIPR_DDA_BUF_LEN
SPACE 1
*****
*              CALLING THE CNV INFO SERVICE              *
*****
SPACE 1
CALL   CUN4LINF,PARMAREA
SPACE 1
EXIT   DS    0H
LM     R15,R0,CUN4BIPR_RC_RS ! SET RETURN AND REASON CODE
L      R13,4(R13)             ! RESTORE CALLER'S R13
L      R14,12(R13)            ! RESTORE R14
LM     R1,R12,24(R13)         ! RESTORE R1-R12 (RETAIN
*                                !   R15 AND R0)
BR     R14
SPACE 1
LTORG ,
SPACE 1
*****
*
*              D E C L A R A T I O N                      *
*
*
*              S e c t i o n                              *
*
*
*****

*      CONSTANT CUN4BIPR_LEN IS USED TO ENSURE THAT SUFFICIENT *
*      STORAGE IS OBTAINED FOR THE PARAMETER AREA.           *
*****
SPACE 1
PARMAREA DC (CUN4BIPR_LEN)X'00' ! STORAGE FOR PARAMETER AREA
ADDA     DC  A(DDA)              ! ADDRESS OF DDA

*****
*      CONSTANT CUN4BIPR_DDA_REQ IS USED TO ENSURE THAT SUFFICIENT *
*      STORAGE FOR THE DDA IS OBTAINED.                             *
*****
SPACE 1
DDA      DC (CUN4BIPR_DDA_Req)X'00' ! DDA SIZE
CCSID1   DC  F'1047'                ! CCSID1
CCSID2   DC  F'1208'                ! CCSID2
SAVEAREA DSECT
DC       F'0'                      ! RESERVED
PREVSA   DC  F'0'                  ! ADDRESS OF PREVIOUS SAVEAREA
NEXTSA   DC  F'0'                  ! ADDRESS OF NEXT   SAVEAREA
SAVER14  DC  F'0'
SAVER15  DC  F'0'
SAVER0   DC  F'0'
SAVER1   DC  F'0'
SAVER2   DC  F'0'
SAVER3   DC  F'0'
SAVER4   DC  F'0'
SAVER5   DC  F'0'
SAVER6   DC  F'0'
SAVER7   DC  F'0'
SAVER8   DC  F'0'
SAVER9   DC  F'0'
SAVER10  DC  F'0'
SAVER11  DC  F'0'
SAVER12  DC  F'0'

```

Conversion information service

```

SPACE 1
COPY CUN4BIID
SPACE 1
CUN4BIID DSECT=YES,LIST=YES
SPACE 1
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
END CUNISMA

```

Mapping of parameters for AMODE (31)

The mapping of the parameter areas is supplied by the interface definition file CUNBIIDF. This file is included in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that might be necessary.

Table 44. Mapping of parameters in HLASM for conversion information service AMODE (31)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	360	DWORD	CUNBIPRM	Parameter area
0	(0)	UNSIGNED	4		CUNBIPRM_Version	Structure version number
4	(4)	UNSIGNED	4		CUNBIPRM_Length	Length of structure
8	(8)	UNSIGNED	4		CUNBIPRM_CCSD1	Specify CCSD1
12	(C)	CHARACTER	32	WORD	CUNBIPRM_CCSD1_ES	CCSD1 encoding scheme (ES) information
12	(C)	CHARACTER	2		*	Reserved
14	(E)	UNSIGNED	2		CUNBIPRM_CCSD1_ES_ID	Encoding scheme ID for CCSD1
16	(10)	CHARACTER	28		CUNBIPRM_CCSD1_ES_Name	Encoding scheme name for CCSD1
44	(2C)	CHARACTER	2		CUNBIPRM_CCSD1_ES_Size	Encoding scheme size for CCSD1
44	(2C)	UNSIGNED	1		CUNBIPRM_CCSD1_ES_Size_ Min	Minimum encoding scheme size for CCSD1
45	(2D)	UNSIGNED	1		CUNBIPRM_CCSD1_ES_Size_ Max	Maximum encoding scheme size for CCSD1
46	(2E)	CHARACTER	2		*	Reserved
48	(30)	CHARACTER	64		CUNBIPRM_CCSD1_Description	CCSD1 description

Table 44. Mapping of parameters in HLASM for conversion information service AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
112	(70)	CHARACTER	8		CUNBIPRM_CCSID1_Num_Subs	Number of substitution characters to every code set for CCSID1
112	(70)	UNSIGNED	1		CUNBIPRM_CCSID1_Num_Subs_ SBCS	Number of substitution characters for SBCS
113	(71)	UNSIGNED	1		CUNBIPRM_CCSID1_Num_Subs_ DBCS	Number of substitution characters for DBCS
114	(72)	UNSIGNED	1		CUNBIPRM_CCSID1_Num_Subs_ TBCS	Number of substitution characters for TBCS
115	(73)	UNSIGNED	1		CUNBIPRM_CCSID1_Num_Subs_ QBCS	Number of substitution characters for QBCS
116	(74)	CHARACTER	4		*	Reserved
120	(78)	CHARACTER	24	WORD	CUNBIPRM_CCSID1_Sub_Char	Substitution characters to be used for CCSID1
120	(78)	CHARACTER	2		CUNBIPRM_CCSID1_Sub_Char_ SBCS	SBCS substitution characters for CCSID1
120	(78)	CHARACTER	1		CUNBIPRM_CCSID1_Sub_Char_ SBCS_1	The second substitution character for the SBCS
121	(79)	CHARACTER	1		CUNBIPRM_CCSID1_Sub_Char_ SBCS_2	The first substitution character for the SBCS
122	(7A)	CHARACTER	4		CUNBIPRM_CCSID1_Sub_Char_ DBCS	DBCS substitution characters for CCSID1
122	(7A)	CHARACTER	2		CUNBIPRM_CCSID1_Sub_Char_ DBCS_1	The second substitution character for the DBCS
124	(7C)	CHARACTER	2		CUNBIPRM_CCSID1_Sub_Char_ DBCS_2	The first substitution character for the DBCS
126	(7E)	CHARACTER	6		CUNBIPRM_CCSID1_Sub_Char_ TBCS	TBCS substitution characters for CCSID1
126	(7E)	CHARACTER	3		CUNBIPRM_CCSID1_Sub_Char_ TBCS_1	The second substitution character for the TBCS
129	(81)	CHARACTER	3		CUNBIPRM_CCSID1_Sub_Char_ TBCS_2	The first substitution character for the TBCS
132	(84)	CHARACTER	8		CUNBIPRM_CCSID1_Sub_Char_ QBCS	QBCS substitution characters for CCSID1
132	(84)	CHARACTER	4		CUNBIPRM_CCSID1_Sub_Char_ QBCS_1	The second substitution character for the QBCS

Conversion information service

Table 44. Mapping of parameters in HLASM for conversion information service AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
136	(88)	CHARACTER	4		CUNBIPRM_CCSID1_Sub_Char_ QBCS_2	The first substitution character for the QBCS
140	(8C)	CHARACTER	4		*	Reserved
144	(90)	CHARACTER	4		*	Reserved
148	(94)	ADDRESS	4		CUNBIPRM_CCSID1_subCCSIDs_ Info_Ptr	Optional pointer to CUNBIPRM_CCSID1_ subCCSIDs_Info
152	(98)	UNSIGNED	4		CUNBIPRM_CCSID1_subCCSIDs_ Info_ALET	ALET for CUNBIPRM_CCSID1_ subCCSIDs_Info_Ptr
156	(9C)	UNSIGNED	1		CUNBIPRM_CCSID1_subCCSIDs_ Info_Num	Number of subCCSIDs for CCSID1
157	(9D)	CHARACTER	3		*	Reserved
160	(A0)	UNSIGNED	4		CUNBIPRM_CCSID2	Specify CCSID2
164	(A4)	CHARACTER	32	WORD	CUNBIPRM_CCSID2_ES	CCSID2 encoding scheme (ES) information
164	(A4)	CHARACTER	2		*	Reserved
166	(A6)	UNSIGNED	2		CUNBIPRM_CCSID2_ES_ID	Encoding scheme ID for CCSID2
168	(A8)	CHARACTER	28		CUNBIPRM_CCSID2_ES_Name	Encoding scheme name for CCSID2
196	(C4)	CHARACTER	2		CUNBIPRM_CCSID2_ES_Size	Encoding scheme size for CCSID2
196	(C4)	UNSIGNED	1		CUNBIPRM_CCSID2_ES_Size_ Min	Minimum encoding scheme size for CCSID2
197	(C5)	UNSIGNED	1		CUNBIPRM_CCSID2_ES_Size_ Max	Maximum encoding scheme size for CCSID1
198	(C6)	CHARACTER	2		*	
200	(C8)	CHARACTER	64		CUNBIPRM_CCSID2_Description	
264	(108)	CHARACTER	8		CUNBIPRM_CCSID2_Num_Subs	Number of substitution characters to every code set for CCSID1
264	(108)	UNSIGNED	1		CUNBIPRM_CCSID2_Num_Subs_ SBCS	Number of substitution characters for SBCS
265	(109)	UNSIGNED	1		CUNBIPRM_CCSID2_Num_Subs_ DBCS	Number of substitution characters for DBCS
266	(10A)	UNSIGNED	1		CUNBIPRM_CCSID2_Num_Subs_ TBCS	Number of substitution characters for TBCS

Table 44. Mapping of parameters in HLASM for conversion information service AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
267	(10B)	UNSIGNED	1		CUNBIPRM_CCSID2_Num_Sub_ QBCS	Number of substitution characters for QBCS
268	(10C)	CHARACTER	4		*	Reserved
272	(110)	CHARACTER	24	WORD	CUNBIPRM_CCSID2_Sub_Char	Substitution characters to be used for CCSID2
272	(110)	CHARACTER	2		CUNBIPRM_CCSID2_Sub_Char_ SBCS	SBCS substitution characters for CCSID2
272	(110)	CHARACTER	1		CUNBIPRM_CCSID2_Sub_Char_ SBCS_1	The second substitution character for the SBCS
273	(111)	CHARACTER	1		CUNBIPRM_CCSID2_Sub_Char_ SBCS_2	The first substitution character for the SBCS
274	(112)	CHARACTER	4		CUNBIPRM_CCSID2_Sub_Char_ DBCS	DBCS substitution characters for CCSID2
274	(112)	CHARACTER	2		CUNBIPRM_CCSID2_Sub_Char_ DBCS_1	The second substitution character for the DBCS
276	(114)	CHARACTER	2		CUNBIPRM_CCSID2_Sub_Char_ DBCS_2	The first substitution character for the DBCS
278	(116)	CHARACTER	6		CUNBIPRM_CCSID2_Sub_Char_ TBCS	TBCS substitution characters for CCSID2
278	(116)	CHARACTER	3		CUNBIPRM_CCSID2_Sub_Char_ TBCS_1	The second substitution character for the TBCS
281	(119)	CHARACTER	3		CUNBIPRM_CCSID2_Sub_Char_ TBCS_2	The first substitution character for the TBCS
284	(11C)	CHARACTER	8		CUNBIPRM_CCSID2_Sub_Char_ QBCS	QBCS substitution characters for CCSID2
284	(11C)	CHARACTER	4		CUNBIPRM_CCSID2_Sub_Char_ QBCS_1	The second substitution character for the QBCS
288	(120)	CHARACTER	4		CUNBIPRM_CCSID2_Sub_Char_ QBCS_2	The first substitution character for the QBCS
292	(124)	CHARACTER	4		*	Reserved
296	(128)	CHARACTER	4		*	Reserved
300	(12C)	ADDRESS	4		CUNBIPRM_CCSID2_ subCCSIDs_Info_Ptr	Optional pointer to CUNBIPRM_CCSID2_ subCCSIDs_Info

Conversion information service

Table 44. Mapping of parameters in HLASM for conversion information service AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
304	(130)	UNSIGNED	4		CUNBIPRM_CCSID2_ subCCSIDs_Info_ALET	ALET for CUNBIPRM_CCSID1_ subCCSIDs_Info_Ptr
308	(134)	UNSIGNED	1		CUNBIPRM_CCSID2_ subCCSIDs_Info_Num	Number of subCCSIDs for CCSID1
309	(135)	CHARACTER	3		*	Reserved
312	(138)	BITSTRING	1		CUNBIPRM_Gen_Flags_Out	Out-FLAG Byte 1 (Set by the service)
		1...			CUNBIPRM_CCSID1_Supported	CCSID1 supported: 0=CCSID1 is not supported. 1=CCSID1 is supported Meaningful if only CCSID1 is provided.
		.1..			CUNBIPRM_CCSID2_Supported	CCSID2 supported: 0=CCSID2 is not supported. 1=CCSID2 is supported. Meaningful if only CCSID2 is provided.
		..1.			CUNBIPRM_Conversion_Supported	Conversion from CCSID1 to CCSID2 is supported: 0=No 1=Yes Meaningful if both CCSID1 and CCSID2 are provided
313	(139)	BITSTRING	1		CUNBIPRM_Gen_Flags_In	In-FLAG Byte 2 (Set by caller)
		1...	..		CUNBIPRM_Get_Tech_Support_ fCCSID2_tCCSID1	Get techniques supported from CCSID2 to CCSID1: 0=Do not obtain techniques. 1=Obtain techniques.
314	(13A)	CHARACTER	6		*	Reserved.
320	(140)	CHARACTER	8		CUNBIPRM_Conv_Tech_ fCCSID1_tCCSID2	Conversion techniques is supported from CCSID1 to CCSID2. Meaningful when Conversion_Supported is turned on.
328	(148)	CHARACTER	8		CUNBIPRM_Conv_Tech_ fCCSID2_tCCSID1	Conversion techniques is supported from CCSID2 to CCSID1. It is meaningful when Conversion_Supported is turned on.

Table 44. Mapping of parameters in HLASM for conversion information service AMODE (31) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
336	(150)	CHARACTER	4		*	Reserved
340	(154)	ADDRESS	4		CUNBIPRM_DDA_Buf_Ptr	Dynamic data area pointer
344	(158)	UNSIGNED	4		CUNBIPRM_DDA_Buf_ALET	Dynamic data area ALET
348	(15C)	UNSIGNED	4		CUNBIPRM_DDA_Buf_Len	Dynamic data area length
352	(160)	CHARACTER	8	WORD	CUNBIPRM_RC_RS	Return/reason code
352	(160)	UNSIGNED	4		CUNBIPRM_Return_Code	Return code
356	(164)	UNSIGNED	4		CUNBIPRM_Reason_Code	Reason code
360	(168)	CHARACTER	0		CUNBIPRM_End	End of CUNBIPRM

Description of parameters in area CUNBIPRM

This description applies to C and HLASM.

CUNBIPRM_Version - set by caller

The default value for CUNBIPRM_Version is CUNBIPRM_Ver, provided in the interface definition file CUNBIIDF. From other values different from CUNBIPRM_Ver (1), z/OS conversion information service returns with CUNBIPRM_Return_Code = CUN_RC_USER_ERR, CUNBIPRM_Reason_Code = CUN_RS_PARM_VER.

CUNBIPRM_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field using the constant CUNBIPRM_Len, which is supplied by the interface definition file CUNBIIDF.

CUNBIPRM_CCSID1 - set by caller, updated by the service

Specifies the CCSID1. This is a numeric four byte field. If this field is not filled out, the rest of the fields with the prefix CUNBIPRM_CCSID1_ are not meaningful after calling the service.

This field is updated by the service when CCSID 1200 is specified and returns the latest Unicode versions available for conversion between CCSID1 and CCSID2. The z/OS Unicode conversion information service updates this field only when both CCSIDs are provided. For individual CCSIDs requests, CUNBIPRM_CCSID1 remains unchanged even when CCSID 1200 is specified.

CUNBIPRM_CCSID1_ES - set by the service

Specifies the encoding scheme (ES) information in the following fields:

CUNBIPRM_CCSID1_ES_ID - set by the service

Specifies the encoding scheme ID for the specified CCSID1.

CUNBIPRM_CCSID1_ES_Name - set by the service

Specifies the encoding scheme name for the specified CCSID1.

The following table shows the ES IDs and the ES names.

ES ID Hex	ES name
1100	EBCDIC, single byte character set (SBCS)
2100	ASCII, SBCS

ES ID Hex	ES name
3100	ASCII, SBCS
4100	ASCII, SBCS
4105	ASCII, SBCS
4155	ASCII, SBCS
5100	ASCII, SBCS
5150	ASCII, SBCS
6100	EBCDIC, SBCS
8100	ASCII, SBCS
1200	EBCDIC, double byte character set (DBCS)
2200	ASCII, DBCS
3200	ASCII, DBCS
5200	ASCII, DBCS
7200	UTF-16 BE
8200	UCS-2
9200	ASCII, DBCS
1301	EBCDIC, Mixed SBCS, DBCS
2300	ASCII, Mixed SBCS, DBCS
2305	ASCII, Mixed SBCS, DBCS
3300	ASCII, Mixed SBCS, DBCS
4403	ASCII
5404	ASCII
5409	ASCII
540A	ASCII
5700	ASCII, triple byte character set (TBCS)
7807	UTF-8, UCS-2 transform
2900	ASCII, quadruple byte character set (QBCS)
2A00	ASCII, Mixed SBCS, DBCS,QBCS
7500	UTF-32 BE
720B	UTF-16 LE

For more information about encoding schemes, see Character Data Representation Architecture Reference (Chapter 3, 'CDRA Identifiers').

CUNBIPRM_CCSD1_ES_Size- set by the service

Specifies the encoding scheme (ES) for the CCSID1. If the ES for CCSID1 supports mixed character set (CS), CUNBIPRM_CCSD1_ES_Size_Min and CUNBIPRM_CCSD1_ES_Size_Max contain different values; otherwise, they contain the same value.

CUNBIPRM_CCSD1_ES_Size_Min - set by the service

Specifies the minimum character set byte size for CCSID1.

CUNBIPRM_CCSD1_ES_Size_Max - set by the service

Specifies the maximum character set byte size for CCSID1.

CUNBIPRM_CCSD1_Description - set by the service

Specifies the description of CCSID1.

CUNBIPRM_CCSID1_Num_Subs - set by the service

Specifies the number of substitution characters to every code set involved by CCSID1.

CUNBIPRM_CCSID1_Num_Subs_SBCS - set by the service

Specifies the number of substitution characters to the SBCS that are involved by CCSID1.

CUNBIPRM_CCSID1_Num_Subs_DBCS - set by the service

Specifies the number of substitution characters to the DBCS that are involved by CCSID1.

CUNBIPRM_CCSID1_Num_Subs_TBCS - set by the service

Specifies the number of substitution characters to the TBCS that are involved by CCSID1.

CUNBIPRM_CCSID1_Num_Subs_QBCS - set by the service

Specifies the number of substitution characters to the QBCS that are involved by CCSID1.

CUNBIPRM_CCSID1_Sub_Char - set by the service

Specifies the substitution character that is to be used for CCSID1. If CCSID1 is specified and the call to the z/OS Unicode conversion information service is successful (CUNBIPRM_CCSID1_Supprtd = 1), the following fields might contain the substitution character for single CCSID or subCCSID involved on CCSID1 (if it is MBCS CCSID).

CUNBIPRM_CCSID1_Sub_Char_SBCS - set by the service

Specifies a SBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve SBCS.

CUNBIPRM_CCSID1_Sub_Char_SBCS_1 - set by the service

Specifies the second substitution character for SBCS. Meaningful if CUNBIPRM_CCSID1_Num_Subs_SBCS is equal to 2.

CUNBIPRM_CCSID1_Sub_Char_SBCS_2 - set by the service

Specifies the first substitution character for the SBCS. Meaningful if CUNBIPRM_CCSID1_Num_Subs_SBCS is equal to 1 or 2.

CUNBIPRM_CCSID1_Sub_Char_DBCS - set by the service

Specifies a DBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve DBCS.

CUNBIPRM_CCSID1_Sub_Char_DBCS_1 - set by the service

Specifies the second substitution character for the DBCS. Meaningful if CUNBIPRM_CCSID1_Num_Subs_DBCS is equal to 2.

CUNBIPRM_CCSID1_Sub_Char_DBCS_2 - set by the service

Specifies the first substitution character for the DBCS. Meaningful if CUNBIPRM_CCSID1_Num_Subs_DBCS is equal to 1 or 2.

CUNBIPRM_CCSID1_Sub_Char_TBCS - set by the service

Specifies a TBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve TBCS.

CUNBIPRM_CCSID1_Sub_Char_TBCS_1 - set by the service

Specifies the second substitution character for the TBCS.
Meaningful if CUNBIPRM_CCSID1_Num_Subs_TBCS is equal to 2.

CUNBIPRM_CCSID1_Sub_Char_TBCS_2 - Set by the service

Specifies the first substitution character for the TBCS.
Meaningful if CUNBIPRM_CCSID1_Num_Subs_TBCS is equal to 1 or 2.

CUNBIPRM_CCSID1_Sub_Char_QBCS - set by the service

Specifies a QBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve QBCS.

CUNBIPRM_CCSID1_Sub_Char_QBCS_1 - set by the service

Specifies the second substitution character for the QBCS.
Meaningful if CUNBIPRM_CCSID1_Num_Subs_QBCS is equal to 2.

CUNBIPRM_CCSID1_Sub_Char_QBCS_2 - set by the service

Specifies the first substitution character for the QBCS.
Meaningful if CUNBIPRM_CCSID1_Num_Subs_QBCS is equal to 1 or 2.

CUNBIPRM_CCSID1_subCCSIDs_Info_Ptr (optional) - set by caller

Specifies an optional additional buffer where z/OS Unicode conversion service information service retrieves information for all of those subCCSIDs for CCSID1. If CCSID1 is not a mixed CCSID, z/OS Unicode conversion service information service does not add anything to this buffer.

IBM recommends that when CUNBIPRM_CCSID1_subCCSIDs_Info_Ptr is specified, verify the contents of CUNBIPRM_CCSID1_subCCSIDs_Info_Num after the service is called successfully.

- If CUNBIPRM_CCSID1_subCCSIDs_Info_Num < 0 or CUNBIPRM_CCSID1_subCCSIDs_Info_Num > 0, CCSID1 is a mixed CCSID. CUNBIPRM_subCCSIDs_Info can be addressed by CUNBIPRM_CCSID1_subCCSIDs_Info_Ptr and CUNBIPRM_CCSID1_subCCSIDs_Info_ALET making a loop CUNBIPRM_CCSID1_subCCSIDs_Info_Num times by the length of CUNBIPRM_subCCSIDs_Info in order to obtain information for the different subCCSIDs that belong to mixed CCSID1.
- Otherwise, CCSID1 is not a mixed conversion.

Also, the size of this buffer must be allocated according to the content of CUNBIPRM_subCCSIDs_Info_Len_Req in a double-word boundary area. CUNBIPRM_subCCSIDs_Info_Len_Req is provided in the IDF file CUNBIIDF.

CUNBIPRM_CCSID1_subCCSIDs_Info_ALET- set by caller

Specifies the alet for CUNBIPRM_CCSID1_subCCSIDs_Info_Ptr and is required if CUNBIPRM_CCSID1_subCCSIDs_Info_Ptr is specified only.

CUNBIPRM_CCSID1_subCCSIDs_Info_Num - set by the service

Specifies the number of subCCSIDs that belong to CCSID1. If CUNBIPRM_CCSID1_subCCSIDs_Info_Num is equal to zero, CCSID1 is not a mixed conversion; otherwise, CCSID1 is a mixed CCSID.

CUNBIPRM_CCSD2- set by the caller, updated by the service

Specifies the CCSD2. This is a numeric four byte field. If this field is not filled out, the rest of the fields with the prefix CUNBIPRM_CCSD2_ are not meaningful after the service is called.

This field is updated by the service when CCSD 1200 is specified, returning the latest Unicode versions available for conversion between CCSD1 and CCSD2. z/OS Unicode conversion information service updates this field only when both CCSDs are provided. For individual CCSD requests, *CUNBIPRM_CCSD2* remains unchanged even when CCSD 1200 is specified.

CUNBIPRM_CCSD2_ES - set by the service

Specifies the ES information in the following fields:

CUNBIPRM_CCSD2_ES_ID - set by the service

Specifies the ES ID for the specified CCSD2.

CUNBIPRM_CCSD1_ES_Name - set by the service

Specifies the ES name for the specified CCSD2.

The following table shows the ES IDs and the ES names.

ES ID Hex	ES name
1100	EBCDIC, single byte character set (SBCS)
2100	ASCII, SBCS
3100	ASCII, SBCS
4100	ASCII, SBCS
4105	ASCII, SBCS
4155	ASCII, SBCS
5100	ASCII, SBCS
5150	ASCII, SBCS
6100	EBCDIC, SBCS
8100	ASCII, SBCS
1200	EBCDIC, double byte character set (DBCS)
2200	ASCII, DBCS
3200	ASCII, DBCS
5200	ASCII, DBCS
7200	UTF-16 BE
8200	UCS-2
9200	ASCII, DBCS
1301	EBCDIC, Mixed SBCS, DBCS
2300	EBCDIC, Mixed SBCS, DBCS
2305	EBCDIC, Mixed SBCS, DBCS
3300	ASCII, Mixed SBCS, DBCS
4403	ASCII
5404	ASCII
5409	ASCII
540A	ASCII

ES ID Hex	ES name
5700	ASCII, triple byte character set (TBCS)
7807	UTF-8, UCS-2 transform
2900	ASCII, quadruple byte character set (QBCS)
2A00	ASCII, Mixed SBCS, DBCS,QBCS
7500	UTF-32 BE
720B	UTF-16 LE

For more information about encoding schemes, see Character Data Representation Architecture Reference (Chapter 3, 'CDRA Identifiers').

CUNBIPRM_CCSID2_ES_Size- set by the service

Specifies the ES (encoding scheme) for the CCSID2. If the ES for CCSID2 supports mixed CS (character set), CUNBIPRM_CCSID2_ES_Size_Min and CUNBIPRM_CCSID2_ES_Size_Max contain different values; otherwise, they contain the same value.

CUNBIPRM_CCSID2_ES_Size_Min - set by the service

Specifies the minimum character set byte size for CCSID2.

CUNBIPRM_CCSID2_ES_Size_Max - set by the service

Specifies the maximum character set byte size for CCSID2.

CUNBIPRM_CCSID2_Description - set by the service

Specifies the description of the CCSID2.

CUNBIPRM_CCSID2_Num_Subs - set by the service

Specifies the number of substitution characters to every code set involved by CCSID2.

CUNBIPRM_CCSID2_Num_Subs_SBCS - set by the service

Specifies the number of substitution characters to the SBCS that are involved by CCSID2.

CUNBIPRM_CCSID2_Num_Subs_DBCS - set by the service

Specifies the number of substitution characters to the DBCS that are involved by CCSID2.

CUNBIPRM_CCSID2_Num_Subs_TBCS - set by the service

Specifies the number of substitution characters to the TBCS that are involved by CCSID2.

CUNBIPRM_CCSID2_Num_Subs_QBCS - set by the service

Specifies the number of substitution characters to the QBCS that are involved by CCSID2.

CUNBIPRM_CCSID2_Sub_Char - set by the service

Specifies the substitution character that is to be used for CCSID2. If CCSID2 is specified and the call to the z/OS Unicode conversion information service is successful (CUNBIPRM_CCSID2_Supprtd = 1), the following fields might contain the substitution character for single CCSID or subCCSID involved in CCSID2 (if it is MBCS CCSID).

CUNBIPRM_CCSID2_Sub_Char_SBCS - set by the service

Specifies a SBCS substitution character for CCSID2. If zero exists, ES for CCSID2 does not involve SBCS.

CUNBIPRM_CCSID2_Sub_Char_SBCS_1 - set by the service

Specifies the second substitution character for the SBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_SBCS is equal to 2.

CUNBIPRM_CCSID2_Sub_Char_SBCS_2 - set by the service

Specifies the first substitution character for the SBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_SBCS is equal to 1 or 2.

CUNBIPRM_CCSID2_Sub_Char_DBCS - set by the service

Specifies a DBCS substitution character for CCSID2. If zero exists, ES for CCSID2 does not involve DBCS.

CUNBIPRM_CCSID2_Sub_Char_DBCS_1 - set by the service

Specifies the second substitution character for the DBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_DBCS is equal to 2.

CUNBIPRM_CCSID2_Sub_Char_DBCS_2 - set by the service

Specifies the first substitution character for the DBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_DBCS is equal to 1 or 2.

CUNBIPRM_CCSID2_Sub_Char_TBCS - set by the service

Specifies a TBCS substitution character for CCSID2. If zero exists, ES for CCSID1 does not involve TBCS.

CUNBIPRM_CCSID2_Sub_Char_TBCS_1 - set by the service

Specifies the second substitution character for the TBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_TBCS is equal to 2.

CUNBIPRM_CCSID2_Sub_Char_TBCS_2 - Set by the service

Specifies the first substitution character for the TBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_TBCS is equal to 1 or 2.

CUNBIPRM_CCSID2_Sub_Char_QBCS - set by the service

Specifies a QBCS substitution character for CCSID2. If zero exists, ES for CCSID2 does not involve QBCS.

CUNBIPRM_CCSID2_Sub_Char_QBCS_1 - set by the service

Specifies the second substitution character for the QBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_QBCS is equal to 2.

CUNBIPRM_CCSID2_Sub_Char_QBCS_2 - set by the service

Specifies the first substitution character for the QBCS.
Meaningful if CUNBIPRM_CCSID2_Num_Subs_QBCS is equal to 1 or 2.

CUNBIPRM_CCSID2_subCCSIDs_Info_Ptr (optional) - set by caller

Specifies an optional additional buffer where z/OS Unicode conversion service information service retrieves information for all of those subCCSIDs for CCSID1. If CCSID2 is not a mixed CCSID, z/OS Unicode conversion service information service does not add anything to this buffer.

IBM recommends that when CUNBIPRM_CCSID2_subCCSIDs_Info_Ptr is specified, verify the contents of CUNBIPRM_CCSID2_subCCSIDs_Info_Num after the service is called successfully.

- If CUNBIPRM_CCSID2_subCCSIDs_Info_Num < 0 or CUNBIPRM_CCSID2_subCCSIDs_Info_Num > 0, CCSID2 is a mixed CCSID. CUNBIPRM_subCCSIDs_Info can be addressed by CUNBIPRM_CCSID2_subCCSIDs_Info_Ptr and CUNBIPRM_CCSID2_subCCSIDs_Info_ALET making a loop CUNBIPRM_CCSID2_subCCSIDs_Info_Num times by the length of CUNBIPRM_subCCSIDs_Info in order to obtain information for the different subCCSIDs that belong to mixed CCSID2.
- Or else, CCSID2 is not a mixed conversion.

Also, the size of this buffer must be allocated according to the content of CUNBIPRM_subCCSIDs_Info_Len_Req in a double-word boundary area. CUNBIPRM_subCCSIDs_Info_Len_Req is provided in the IDF file CUNBIIDF.

CUNBIPRM_CCSID2_subCCSIDs_Info_ALET- set by caller

Specifies the alet for CUNBIPRM_CCSID2_subCCSIDs_Info_Ptr and is required if CUNBIPRM_CCSID2_subCCSIDs_Info_Ptr is specified only.

CUNBIPRM_CCSID2_subCCSIDs_Info_Num - set by the service

Specifies the number of subCCSIDs that belong to CCSID2. If CUNBIPRM_CCSID1_subCCSIDs_Info_Num is equal to zero, CCSID2 is not a mixed conversion; otherwise, CCSID2 is a mixed CCSID.

CUNBIPRM_Gen_Flags_Out - set by the service

Specifies output results from the z/OS Unicode conversion information service according to the description of the following bit fields.

CUNBIPRM_CCSID1_Supported - set by the service

Specifies whether CCSID1 information is retrieved successfully after calling the z/OS Unicode conversion information service, according to the following values:

- 0** CCSID1 is not supported.
- 1** CCSID1 is supported.

CUNBIPRM_CCSID1_Supported is meaningful when CCSID1 is provided.

CUNBIPRM_CCSID2_Supported - set by the service

Specifies whether CCSID2 information is retrieved successfully after calling the z/OS Unicode conversion information service, according to the following values:

- 0** CCSID2 is not supported.
- 1** CCSID2 is supported.

CUNBIPRM_CCSID2_Supported is meaningful when CCSID2 is provided.

CUNBIPRM_Conversion_Supported - set by the service

Specifies whether the conversion between CCSIDs provided by CUNBIPRM_CCSID1 and CUNBIPRM_CCSID2 are supported, according the following values:

- 0** Conversion is not supported.
- 1** Conversion is supported.

CUNBIPRM_Conversion_Supported is meaningful when both CCSID1 and CCSID2 are provided.

CUNBIPRM_Gen_Flags_In - set by caller**CUNBIPRM_Get_Tech_Supp_fCCSID2_tCCSID1 -set by caller**

Specifies whether techniques supported for CCSID2 to CCSID1 are returned at CUNBIPRM_Conv_Tech_fCCSID2_tCCSID1, according the following values:

- 0** Do not obtain techniques supported from CCSID2 to CCSID1. This is the default.
- 1** Obtain techniques supported from CCSID2 to CCSID1.

CUNBIPRM_Conv_Tech_fCCSID1_tCCSID2- set by the service

Specifies the conversion techniques supported for CCSID1 to CCSID2. CUNBIPRM_Conv_Tech_fCCSID1_tCCSID2 is meaningful when CUNBIPRM_Conversion_Supported is on.

CUNBIPRM_Conv_Tech_fCCSID2_tCCSID1- set by the service

Specifies the conversion techniques supported for CCSID2 to CCSID1. CUNBIPRM_Conv_Tech_fCCSID2_tCCSID1 is meaningful when CUNBIPRM_Conversion_Supported is on.

CUNBIPRM_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion information services are using internally as dynamic data area.

Note: CUNBIPRM_DDA_Buf_Ptr must be in a double-word boundary area.

CUNBIPRM_DDA_Buf_ALET - set by caller

Specifies the alet to be used if the dynamic data area addressed by CUNBIPRM_DDA_Buf_Ptr resides in a different address or data space.

CUNBIPRM_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by CUNBIPRM_DDA_Buf_Ptr. The required length is defined by constant CUNBIPRM_DDA_Req provided in the interface definition file CUNBIIDF.

CUNBIPRM_RC_RS - set by the service

Specifies the return code and reason code.

CUNBIPRM_Return_Code - set by the service

Specifies the return code.

CUNBIPRM_Reason_Code - set by the service

Specifies the reason code.

CUNBIPRM_subCCSIDs_CCSID - set by the service

Specifies a subCCSIDs.

CUNBIPRM_subCCSIDs_Size - set by the service

Specifies the size character for the subCCSID.

CUNBIPRM_subCCSIDs_Description - set by the service

Specifies the description of the subCCSID.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas is supplied by the interface definition file CUN4BIID. This file is included in the SYS1.MACLIB data set, and contains the length of each parameter and any boundary alignment that might be necessary.

Mapping of parameters for AMODE (64)

Table 45. Mapping of parameters in HLASM for conversion information service AMODE (64)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
0	(0)	STRUCTURE	360	DWORD	CUN4BIPR	Parameter area
0	(0)	UNSIGNED	4		CUN4BIPR_Version	Structure version number
4	(4)	UNSIGNED	4		CUN4BIPR_Length	Length of structure
8	(8)	UNSIGNED	4		CUN4BIPR_CCSID1	Specify CCSID1
12	(C)	CHARACTER	32	WORD	CUN4BIPR_CCSID1_ES	CCSID1 encoding scheme (ES) information
12	(C)	CHARACTER	2		*	Reserved
14	(E)	UNSIGNED	2		CUN4BIPR_CCSID1_ES_ID	Encoding scheme ID for CCSID1
16	(10)	CHARACTER	28		CUN4BIPR_CCSID1_ES_Name	Encoding scheme name for CCSID1
44	(2C)	CHARACTER	2		CUN4BIPR_CCSID1_ES_Size	Encoding scheme size for CCSID1
44	(2C)	UNSIGNED	1		CUN4BIPR_CCSID1_ES_Size_Min	Minimum encoding scheme size for CCSID1
45	(2D)	UNSIGNED	1		CUN4BIPR_CCSID1_ES_Size_Max	Maximum encoding scheme size for CCSID1
46	(2E)	CHARACTER	2		*	Reserved
48	(30)	CHARACTER	64		CUN4BIPR_CCSID1_Description	CCSID1 description
112	(70)	CHARACTER	8		CUN4BIPR_CCSID1_Num_Subs	Number of substitution characters to every code set for CCSID1
112	(70)	UNSIGNED	1		CUN4BIPR_CCSID1_Num_Subs_SBCS	Number of substitution characters for SBCS
113	(71)	UNSIGNED	1		CUN4BIPR_CCSID1_Num_Subs_DBCS	Number of substitution characters for DBCS
114	(72)	UNSIGNED	1		CUN4BIPR_CCSID1_Num_Subs_TBCS	Number of substitution characters for TBCS
115	(73)	UNSIGNED	1		CUN4BIPR_CCSID1_Num_Subs_QBCS	Number of substitution characters for QBCS
116	(74)	CHARACTER	4		*	Reserved
120	(78)	CHARACTER	24	WORD	CUN4BIPR_CCSID1_Sub_Char	Substitution characters to be used for CCSID1
120	(78)	CHARACTER	2		CUN4BIPR_CCSID1_Sub_Char_SBCS	SBCS substitution characters for CCSID1
120	(78)	CHARACTER	1		CUN4BIPR_CCSID1_Sub_Char_SBCS_1	The second substitution character for the SBCS

Mapping of parameters for AMODE (64)

Table 45. Mapping of parameters in HLASM for conversion information service AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
121	(79)	CHARACTER	1		CUN4BIPR_CCSID1_Sub_Char_SBCS_2	The first substitution character for the SBCS
122	(7A)	CHARACTER	4		CUN4BIPR_CCSID1_Sub_Char_DBCS	DBCS substitution characters for CCSID1
122	(7A)	CHARACTER	2		CUN4BIPR_CCSID1_Sub_Char_DBCS_1	The second substitution character for the DBCS
124	(7C)	CHARACTER	2		CUN4BIPR_CCSID1_Sub_Char_DBCS_2	The first substitution character for the DBCS
126	(7E)	CHARACTER	6		CUN4BIPR_CCSID1_Sub_Char_TBCS	TBCS substitution characters for CCSID1
126	(7E)	CHARACTER	3		CUN4BIPR_CCSID1_Sub_Char_TBCS_1	The second substitution character for the TBCS
129	(81)	CHARACTER	3		CUN4BIPR_CCSID1_Sub_Char_TBCS_2	The first substitution character for the TBCS
132	(84)	CHARACTER	8		CUN4BIPR_CCSID1_Sub_Char_QBCS	QBCS substitution characters for CCSID1
132	(84)	CHARACTER	4		CUN4BIPR_CCSID1_Sub_Char_QBCS_1	The second substitution character for the QBCS
136	(88)	CHARACTER	4		CUN4BIPR_CCSID1_Sub_Char_QBCS_2	The first substitution character for the QBCS
140	(8C)	CHARACTER	4		*	Reserved
144	(90)	CHARACTER	4		*	Reserved
148	(94)	ADDRESS	4		CUN4BIPR_CCSID1_subCCSIDs_Info_Ptr	Optional pointer to CUN4BIPR_CCSID1_subCCSIDs_Info
152	(98)	UNSIGNED	4		CUN4BIPR_CCSID1_subCCSIDs_Info_ALET	ALET for CUN4BIPR_CCSID1_subCCSIDs_Info_Ptr
156	(9C)	UNSIGNED	1		CUN4BIPR_CCSID1_subCCSIDs_Info_Num	Number of subCCSIDs for CCSID1
157	(9D)	CHARACTER	3		*	Reserved
160	(A0)	UNSIGNED	4		CUN4BIPR_CCSID2	Specify CCSID2
164	(A4)	CHARACTER	32	WORD	CUN4BIPR_CCSID2_ES	CCSID2 encoding scheme (ES) information
164	(A4)	CHARACTER	2		*	Reserved

Mapping of parameters for AMODE (64)

Table 45. Mapping of parameters in HLASM for conversion information service AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
166	(A6)	UNSIGNED	2		CUN4BIPR_CCSID2_ES_ID	Encoding scheme ID for CCSID2
168	(A8)	CHARACTER	28		CUN4BIPR_CCSID2_ES_Name	Encoding scheme name for CCSID2
196	(C4)	CHARACTER	2		CUN4BIPR_CCSID2_ES_Size	Encoding scheme size for CCSID2
196	(C4)	UNSIGNED	1		CUN4BIPR_CCSID2_ES_Size_ Min	Minimum encoding scheme size for CCSID2
197	(C5)	UNSIGNED	1		CUN4BIPR_CCSID2_ES_Size_ Max	Maximum encoding scheme size for CCSID1
198	(C6)	CHARACTER	2		*	
200	(C8)	CHARACTER	64		CUN4BIPR_CCSID2_Description	
264	(108)	CHARACTER	8		CUN4BIPR_CCSID2_Num_Subs	Number of substitution characters to every code set for CCSID1
264	(108)	UNSIGNED	1		CUN4BIPR_CCSID2_Num_Subs_ SBCS	Number of substitution characters for SBCS
265	(109)	UNSIGNED	1		CUN4BIPR_CCSID2_Num_Subs_ DBCS	Number of substitution characters for DBCS
266	(10A)	UNSIGNED	1		CUN4BIPR_CCSID2_Num_Subs_ TBCS	Number of substitution character for TBCS
267	(10B)	UNSIGNED	1		CUN4BIPR_CCSID2_Num_Subs_ QBCS	Number of substitution characters for QBCS
268	(10C)	CHARACTER	4		*	Reserved
272	(110)	CHARACTER	24	WORD	CUN4BIPR_CCSID2_Sub_Char	Substitution characters to be used for CCSID2
272	(110)	CHARACTER	2		CUN4BIPR_CCSID2_Sub_Char_ SBCS	SBCS substitution characters for CCSID2
272	(110)	CHARACTER	1		CUN4BIPR_CCSID2_Sub_Char_ SBCS_1	The second substitution character for the SBCS
273	(111)	CHARACTER	1		CUN4BIPR_CCSID2_Sub_Char_ SBCS_2	The first substitution character for the SBCS
274	(112)	CHARACTER	4		CUN4BIPR_CCSID2_Sub_Char_ DBCS	DBCS substitution characters for CCSID2
274	(112)	CHARACTER	2		CUN4BIPR_CCSID2_Sub_Char_ DBCS_1	The second substitution character for the DBCS

Mapping of parameters for AMODE (64)

Table 45. Mapping of parameters in HLASM for conversion information service AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
276	(114)	CHARACTER	2		CUN4BIPR_CCSID2_Sub_Char_ DBCS_2	The first substitution character for the DBCS
278	(116)	CHARACTER	6		CUN4BIPR_CCSID2_Sub_Char_ TBCS	TBCS substitution characters for CCSID2
278	(116)	CHARACTER	3		CUN4BIPR_CCSID2_Sub_Char_ TBCS_1	The second substitution character for the TBCS
281	(119)	CHARACTER	3		CUN4BIPR_CCSID2_Sub_Char_ TBCS_2	The first substitution character for the TBCS
284	(11C)	CHARACTER	8		CUN4BIPR_CCSID2_Sub_Char_ QBCS	QBCS substitution characters for CCSID2
284	(11C)	CHARACTER	4		CUN4BIPR_CCSID2_Sub_Char_ QBCS_1	The second substitution character for the QBCS
288	(120)	CHARACTER	4		CUN4BIPR_CCSID2_Sub_Char_ QBCS_2	The first substitution character for the QBCS
292	(124)	CHARACTER	4		*	Reserved
296	(128)	CHARACTER	4		*	Reserved
300	(12C)	ADDRESS	4		CUN4BIPR_CCSID2_ subCCSIDs_Info_Ptr	Optional pointer to CUN4BIPR_CCSID2_ subCCSIDs_Info
304	(130)	UNSIGNED	4		CUN4BIPR_CCSID2_ subCCSIDs_Info_ALET	ALET for CUN4BIPR_CCSID1_ subCCSIDs_Info_Ptr
308	(134)	UNSIGNED	1		CUN4BIPR_CCSID2_ subCCSIDs_Info_Num	Number of subCCSIDs for CCSID1
309	(135)	CHARACTER	3		*	Reserved

Mapping of parameters for AMODE (64)

Table 45. Mapping of parameters in HLASM for conversion information service AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
312	(138)	BITSTRING	1		CUN4BIPR_Gen_Flags_Out	Out-FLAG Byte 1 (Set by the service)
		1...			CUN4BIPR_CCSID1_Supported	CCSID1 supported: 0=CCSID1 is not supported. 1=CCSID1 is supported Meaningful if only CCSID1 is provided.
		.1..			CUN4BIPR_CCSID2_Supported	CCSID2 supported: 0=CCSID2 is not supported. 1=CCSID2 is supported. Meaningful if only CCSID2 is provided.
		..1.			CUN4BIPR_Conversion_Supported	Conversion from CCSID1 to CCSID2 is supported: 0=No 1=Yes Meaningful if both CCSID1 and CCSID2 are provided
313	(139)	BITSTRING	1		CUN4BIPR_Gen_Flags_In	In-FLAG Byte 2 (Set by caller)
		1...	..		CUN4BIPR_Get_Tech_Support_ fCCSID2_tCCSID1	Get techniques supported from CCSID2 to CCSID1: 0=Do not obtain techniques. 1=Obtain techniques.
314	(13A)	CHARACTER	6		*	Reserved
320	(140)	CHARACTER	8		CUN4BIPR_Conv_Tech_ fCCSID1_tCCSID2	Conversion techniques is supported from CCSID1 to CCSID2. Meaningful when Conversion_Supported is turned on.
328	(148)	CHARACTER	8		CUN4BIPR_Conv_Tech_ fCCSID2_tCCSID1	Conversion techniques is supported from CCSID2 to CCSID1. It is meaningful when Conversion_Supported is turned on.
336	(150)	CHARACTER	4		*	Reserved
340	(154)	ADDRESS	4		CUN4BIPR_DDA_Buf_Ptr	Dynamic data area pointer
344	(158)	UNSIGNED	4		CUN4BIPR_DDA_Buf_ALET	Dynamic data area ALET
348	(15C)	UNSIGNED	4		CUN4BIPR_DDA_Buf_Len	Dynamic data area length
352	(160)	CHARACTER	8	WORD	CUN4BIPR_RC_RS	Return/reason code
352	(160)	UNSIGNED	4		CUN4BIPR_Return_Code	Return code

Table 45. Mapping of parameters in HLASM for conversion information service AMODE (64) (continued)

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Short Description - See full description following table for details
356	(164)	UNSIGNED	4		CUN4BIPR_Reason_Code	Reason code
360	(168)	CHARACTER	0		CUN4BIPR_End	End of CUN4BIPRM

Description of parameters in area CUN4BIPR

This description applies to C and HLASM.

CUN4BIPR_Version - set by caller

The default value for CUN4BIPR_Version is CUN4BIPR_Ver, provided in the interface definition file CUN4BIID. For other values different from CUN4BIPR_Ver (1), z/OS conversion information service returns with CUN4BIPR_Return_Code = CUN_RC_USER_ERR, CUN4BIPR_Reason_Code = CUN_RS_PARM_VER.

CUN4BIPR_Length - set by caller

Specifies the length of the parameter area. HLASM users must initialize this field using the constant CUN4BIPR_Len that is supplied by the interface definition file CUN4BIID.

CUN4BIPR_CCSD1 - set by caller, updated by the service

Specifies the CCSID1. This is a numeric four byte field. If this field is not filled out, the rest of the fields with the prefix CUN4BIPR_CCSD1_ are not meaningful after calling the service.

This field is updated by the service when CCSID 1200 is specified, returning the latest Unicode versions available for conversion between CCSID1 and CCSID2. The z/OS Unicode conversion information service updates this field only when both CCSIDs are provided. For individual CCSID requests, CUN4BIPR_CCSD1 remains unchanged even when CCSID 1200 is specified.

CUN4BIPR_CCSD1_ES - set by the service

Specifies the encoding scheme (ES) information in the following fields:

CUN4BIPR_CCSD1_ES_ID - set by the service

Specifies the encoding scheme ID for the specified CCSID1.

CUN4BIPR_CCSD1_ES_Name - set by the service

Specifies the encoding scheme name for the specified CCSID1.

The following table shows the ES IDs and the ES names.

ES ID Hex	ES name
1100	EBCDIC, single byte character set (SBCS)
2100	ASCII, SBCS
3100	ASCII, SBCS
4100	ASCII, SBCS
4105	ASCII, SBCS
4155	ASCII, SBCS
5100	ASCII, SBCS
5150	ASCII, SBCS

Mapping of parameters for AMODE (64)

ES ID Hex	ES name
6100	EBCDIC, SBCS
8100	ASCII, SBCS
1200	EBCDIC, double byte character set (DBCS)
2200	ASCII, DBCS
3200	ASCII, DBCS
5200	ASCII, DBCS
7200	UTF-16 BE
8200	UCS-2
9200	ASCII, DBCS
1301	EBCDIC, Mixed SBCS, DBCS
2300	ASCII, Mixed SBCS, DBCS
2305	ASCII, Mixed SBCS, DBCS
3300	ASCII, Mixed SBCS, DBCS
4403	ASCII
5404	ASCII
5409	ASCII
540A	ASCII
5700	ASCII, triple byte character set (TBCS)
7807	UTF-8, UCS-2 transform
2900	ASCII, Quadruple byte character set (QBCS)
2A00	ASCII, Mixed SBCS, DBCS, QBCS
7500	UTF-32 BE
720B	UTF-16 LE

For more information about encoding schemes, see Character Data Representation Architecture Reference (Chapter 3, 'CDRA Identifiers').

CUN4BIPR_CCSID1_ES_Size- set by the service

Specifies the encoding scheme (ES) for the CCSID1. If the ES for CCSID1 supports mixed character set (CS), CUN4BIPR_CCSID1_ES_Size_Min and CUN4BIPR_CCSID1_ES_Size_Max contain different values; otherwise, they contain the same value.

CUN4BIPR_CCSID1_ES_Size_Min - set by the service

Specifies the minimum character set byte size for CCSID1.

CUN4BIPR_CCSID1_ES_Size_Max - set by the service

Specifies the maximum character set byte size for CCSID1.

CUN4BIPR_CCSID1_Description - set by the service

Specifies the description of the CCSID1.

CUN4BIPR_CCSID1_Num_Subs - set by the service

Specifies the number of substitution characters to every code set involved by CCSID1.

CUN4BIPR_CCSID1_Num_Subs_SBCS - set by the service

Specifies the number of substitution characters to the SBCS that are involved by CCSID1.

CUN4BIPR_CCSID1_Num_Sub_DBCS - set by the service

Specifies the number of substitution characters to the DBCS that are involved by CCSID1.

CUN4BIPR_CCSID1_Num_Sub_TBCS - set by the service

Specifies the number of substitution characters to the TBCS that are involved by CCSID1.

CUN4BIPR_CCSID1_Num_Sub_QBCS - set by the service

Specifies the number of substitution characters to the QBCS that are involved by CCSID1.

CUN4BIPR_CCSID1_Sub_Char - set by the service

Specifies the substitution character that is to be used for CCSID1. If CCSID1 is specified and the call to the z/OS Unicode conversion information service is successful (CUN4BIPR_CCSID1_Supported = 1), the following fields might contain the substitution character for single CCSID or subCCSID involved in CCSID1 (if it is MBCS CCSID).

CUN4BIPR_CCSID1_Sub_Char_SBCS - set by the service

Specifies a SBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve SBCS.

CUN4BIPR_CCSID1_Sub_Char_SBCS_1 - set by the service

Specifies the second substitution character for the SBCS. Meaningful if CUN4BIPR_CCSID1_Num_Sub_SBCS is equal to 2.

CUN4BIPR_CCSID1_Sub_Char_SBCS_2 - set by the service

Specifies the first substitution character for the SBCS. Meaningful if CUN4BIPR_CCSID1_Num_Sub_SBCS is equal to 1 or 2.

CUN4BIPR_CCSID1_Sub_Char_DBCS - set by the service

Specifies a DBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve DBCS.

CUN4BIPR_CCSID1_Sub_Char_DBCS_1 - set by the service

Specifies the second substitution character for the DBCS. Meaningful if CUN4BIPR_CCSID1_Num_Sub_DBCS is equal to 2.

CUN4BIPR_CCSID1_Sub_Char_DBCS_2 - set by the service

Specifies the first substitution character for the DBCS. Meaningful if CUN4BIPR_CCSID1_Num_Sub_DBCS is equal to 1 or 2.

CUN4BIPR_CCSID1_Sub_Char_TBCS - set by the service

Specifies a TBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve TBCS.

CUN4BIPR_CCSID1_Sub_Char_TBCS_1 - set by the service

Specifies the second substitution character for the TBCS. Meaningful if CUN4BIPR_CCSID1_Num_Sub_TBCS is equal to 2.

CUN4BIPR_CCSID1_Sub_Char_TBCS_2 - Set by the service

Specifies the first substitution character for the TBCS. Meaningful if CUN4BIPR_CCSID1_Num_Sub_TBCS is equal to 1 or 2.

Mapping of parameters for AMODE (64)

CUN4BIPR_CCSID1_Sub_Char_QBCS - set by the service

Specifies a QBCS substitution character for CCSID1. If zero exists, ES for CCSID1 does not involve QBCS.

CUN4BIPR_CCSID1_Sub_Char_QBCS_1 - set by the service

Specifies the second substitution character for the QBCS. Meaningful if CUN4BIPR_CCSID1_Num_Subs_QBCS is equal to 2.

CUN4BIPR_CCSID1_Sub_Char_QBCS_2 - set by the service

Specifies the first substitution character for the QBCS. Meaningful if CUN4BIPR_CCSID1_Num_Subs_QBCS is equal to 1 or 2.

CUN4BIPR_CCSID1_subCCSIDs_Info_Ptr (optional) - set by caller

Specifies an optional additional buffer where z/OS Unicode conversion service information service retrieves information for all of those subCCSIDs for CCSID1. If CCSID1 is not a mixed CCSID, z/OS Unicode conversion service information service does not add anything to this buffer.

IBM recommends that when CUN4BIPR_CCSID1_subCCSIDs_Info_Ptr is specified, verify the contents of CUN4BIPR_CCSID1_subCCSIDs_Info_Num after the service is called successfully.

- If CUN4BIPR_CCSID1_subCCSIDs_Info_Num < 0 or CUN4BIPR_CCSID1_subCCSIDs_Info_Num > 0, CCSID1 is a mixed CCSID. CUN4BIPR_subCCSIDs_Info can be addressed by CUN4BIPR_CCSID1_subCCSIDs_Info_Ptr and CUN4BIPR_CCSID1_subCCSIDs_Info_ALET making a loop CUN4BIPR_CCSID1_subCCSIDs_Info_Num times by the length of CUN4BIPR_subCCSIDs_Info in order to obtain information for the different subCCSIDs that belong to mixed CCSID1.
- Otherwise, CCSID1 is not a mixed conversion.

Also, the size of this buffer must be allocated according to the content of CUN4BIPR_subCCSIDs_Info_Len_Req in a double-word boundary area. CUN4BIPR_subCCSIDs_Info_Len_Req is provided in the IDF file CUN4BIID.

CUN4BIPR_CCSID1_subCCSIDs_Info_ALET- set by caller

Specifies the alet for CUN4BIPR_CCSID1_subCCSIDs_Info_Ptr and is required if CUN4BIPR_CCSID1_subCCSIDs_Info_Ptr is specified only.

CUN4BIPR_CCSID1_subCCSIDs_Info_Num - set by the service

Specifies the number of subCCSIDs that belong to CCSID1. If CUN4BIPR_CCSID1_subCCSIDs_Info_Num is equal to zero, CCSID1 is not a mixed conversion; otherwise, CCSID1 is a mixed CCSID.

CUN4BIPR_CCSID2- set by the caller, updated by the service

Specifies the CCSID2. This is a numeric four byte field. If this field is not filled out, the rest of the fields with the prefix CUN4BIPR_CCSID2_ are not meaningful after the service is called.

This field is updated by the service when CCSID 1200 is specified, returning the latest Unicode versions available for conversion between CCSID1 and CCSID2. z/OS Unicode conversion information service updates this field only when both two CCSIDs are provided. For individual CCSID requests, CUN4BIPR_CCSID2 remains unchanged even when CCSID 1200 is specified.

CUN4BIPR_CCSID2_ES - set by the service

Specifies the ES information in the following fields:

CUN4BIPR_CCSID2_ES_ID - set by the service

Specifies the ES ID for the specified CCSID2.

CUN4BIPR_CCSID1_ES_Name - set by the service

Specifies the ES name for the specified CCSID2.

The following table shows the ES IDs and the ES names.

ES ID Hex	ES name
1100	EBCDIC, single byte character set (SBCS)
2100	ASCII, SBCS
3100	ASCII, SBCS
4100	ASCII, SBCS
4105	ASCII, SBCS
4155	ASCII, SBCS
5100	ASCII, SBCS
5150	ASCII, SBCS
6100	EBCDIC, SBCS
8100	ASCII, SBCS
1200	EBCDIC, double byte character set (DBCS)
2200	ASCII, DBCS
3200	ASCII, DBCS
5200	ASCII, DBCS
7200	UTF-16 BE
8200	UCS-2
9200	ASCII, DBCS
1301	EBCDIC, Mixed SBCS, DBCS
2300	ASCII, Mixed SBCS, DBCS
2305	ASCII, Mixed SBCS, DBCS
3300	ASCII, Mixed SBCS, DBCS
4403	ASCII
5404	ASCII
5409	ASCII
540A	ASCII
5700	ASCII, triple byte character set (TBCS)
7807	UTF-8, UCS-2 transform
2900	ASCII, quadruple byte character set (QBCS)
2A00	ASCII, Mixed SBCS, DBCS,QBCS
7500	UTF-32 BE
720B	UTF-16 LE

For more information about encoding schemes, see Character Data Representation Architecture Reference (Chapter 3, 'CDRA Identifiers').

Mapping of parameters for AMODE (64)

CUN4BIPR_CCSID2_ES_Size- set by the service

Specifies the ES (encoding scheme) for the CCSID2. If the ES for CCSID2 supports mixed CS (Character set), CUN4BIPR_CCSID2_ES_Size_Min and CUN4BIPR_CCSID2_ES_Size_Max contain different values; otherwise, they contain the same value.

CUN4BIPR_CCSID2_ES_Size_Min - set by the service

Specifies the minimum character set byte size for CCSID2.

CUN4BIPR_CCSID2_ES_Size_Max - set by the service

Specifies the maximum character set byte size for CCSID2.

CUN4BIPR_CCSID2_Description - set by the service

Specifies the description of the CCSID2.

CUN4BIPR_CCSID2_Num_Subs - set by the service

Specifies the number of substitution characters to every code set involved by CCSID2.

CUN4BIPR_CCSID2_Num_Subs_SBCS - set by the service

Specifies the number of substitution characters to the SBCS that are involved by CCSID2.

CUN4BIPR_CCSID2_Num_Subs_DBCS - set by the service

Specifies the number of substitution characters to the DBCS that are involved by CCSID2.

CUN4BIPR_CCSID2_Num_Subs_TBCS - set by the service

Specifies the number of substitution characters to the TBCS that are involved by CCSID2.

CUN4BIPR_CCSID2_Num_Subs_QBCS - set by the service

Specifies the number of substitution characters to the QBCS that are involved by CCSID2.

CUN4BIPR_CCSID2_Sub_Char - set by the service

Specifies the substitution character that is to be used for CCSID2. If CCSID2 is specified and the call to the z/OS Unicode conversion information service is successful (CUN4BIPR_CCSID2_Supprtd = 1), the following fields might contain the substitution character for single CCSID or subCCSID involved in CCSID2 (if it is MBCS CCSID).

CUN4BIPR_CCSID2_Sub_Char_SBCS - set by the service

Specifies a SBCS substitution character for CCSID2. If zero exists, ES for CCSID2 does not involve SBCS.

CUN4BIPR_CCSID2_Sub_Char_SBCS_1 - set by the service

Specifies the second substitution character for the SBCS. Meaningful if CUN4BIPR_CCSID2_Num_Subs_SBCS is equal to 2.

CUN4BIPR_CCSID2_Sub_Char_SBCS_2 - set by the service

Specifies the first substitution character for the SBCS. Meaningful if CUN4BIPR_CCSID2_Num_Subs_SBCS is equal to 1 or 2.

CUN4BIPR_CCSID2_Sub_Char_DBCS - set by the service

Specifies a DBCS substitution character for CCSID2. If zero exists, ES for CCSID2 does not involve DBCS.

CUN4BIPR_CCSID2_Sub_Char_DBCS_1 - set by the service

Specifies the second substitution character for the DBCS.
Meaningful if CUN4BIPR_CCSID2_Num_Subs_DBCS is equal to 2.

CUN4BIPR_CCSID2_Sub_Char_DBCS_2 - set by the service

Specifies the first substitution character for the DBCS.
Meaningful if CUN4BIPR_CCSID2_Num_Subs_DBCS is equal to 1 or 2.

CUN4BIPR_CCSID2_Sub_Char_TBCS - set by the service

Specifies a TBCS substitution character for CCSID2. If zero exists, ES for CCSID1 does not involve TBCS.

CUN4BIPR_CCSID2_Sub_Char_TBCS_1 - set by the service

Specifies the second substitution character for the TBCS.
Meaningful if CUN4BIPR_CCSID2_Num_Subs_TBCS is equal to 2.

CUN4BIPR_CCSID2_Sub_Char_TBCS_2 - Set by the service

Specifies the first substitution character for the TBCS.
Meaningful if CUN4BIPR_CCSID2_Num_Subs_TBCS is equal to 1 or 2.

CUN4BIPR_CCSID2_Sub_Char_QBCS - set by the service

Specifies a QBCS substitution character for CCSID2. If zero exists, ES for CCSID2 does not involve QBCS.

CUN4BIPR_CCSID2_Sub_Char_QBCS_1 - set by the service

Specifies the second substitution character for the QBCS.
Meaningful if CUN4BIPR_CCSID2_Num_Subs_QBCS is equal to 2.

CUN4BIPR_CCSID2_Sub_Char_QBCS_2 - set by the service

Specifies the first substitution character for the QBCS.
Meaningful if CUN4BIPR_CCSID2_Num_Subs_QBCS is equal to 1 or 2.

CUN4BIPR_CCSID2_subCCSIDs_Info_Ptr (optional) - set by caller

Specifies an optional additional buffer where z/OS Unicode conversion service information service retrieves information for all of those subCCSIDs for CCSID1. If CCSID2 is not a mixed CCSID, z/OS Unicode conversion service information service does not add anything to this buffer.

IBM recommends that when CUN4BIPR_CCSID2_subCCSIDs_Info_Ptr is specified, verify the contents of CUN4BIPR_CCSID2_subCCSIDs_Info_Num after calling the service successfully.

- If CUN4BIPR_CCSID2_subCCSIDs_Info_Num < 0 or CUN4BIPR_CCSID2_subCCSIDs_Info_Num > 0, CCSID2 is a mixed CCSID. CUN4BIPR_subCCSIDs_Info can be addressed by CUN4BIPR_CCSID2_subCCSIDs_Info_Ptr and CUN4BIPR_CCSID2_subCCSIDs_Info_ALET making a loop CUN4BIPR_CCSID2_subCCSIDs_Info_Num times by the length of CUN4BIPR_subCCSIDs_Info in order to obtain information for the different subCCSIDs that belong to mixed CCSID2.
- Or else, CCSID2 is not a mixed conversion.

Mapping of parameters for AMODE (64)

Also, the size of this buffer must be allocated according to the content of CUN4BIPR_subCCSIDs_Info_Len_Req in a double-word boundary area. CUN4BIPR_subCCSIDs_Info_Len_Req is provided in the IDF file CUN4BIID.

CUN4BIPR_CCSD2_subCCSIDs_Info_ALET- set by caller

Specifies the alet for CUN4BIPR_CCSD2_subCCSIDs_Info_Ptr and is required if CUN4BIPR_CCSD2_subCCSIDs_Info_Ptr is specified only.

CUN4BIPR_CCSD2_subCCSIDs_Info_Num - set by the service

Specifies the number of subCCSIDs that belong to CCSID2. If CUN4BIPR_CCSD1_subCCSIDs_Info_Num is equal to zero, CCSID2 is not a mixed conversion; otherwise, CCSID2 is a mixed CCSID.

CUN4BIPR_Gen_Flags_Out - set by the service

Specifies output results from the z/OS Unicode conversion information service according to the description of the following bit fields.

CUN4BIPR_CCSD1_Supported - set by the service

Specifies whether CCSID1 information is retrieved successfully after the z/OS Unicode conversion information service is called, according to the following values:

0 CCSID1 is not supported.

1 CCSID1 is supported.

CUN4BIPR_CCSD1_Supported is meaningful when CCSID1 is provided.

CUNBIPR_CCSD2_Supported - set by the service

Specifies whether CCSID2 information is retrieved successfully after the z/OS Unicode conversion information service is called, according to the following values:

0 CCSID2 is not supported.

1 CCSID2 is supported.

CUN4BIPR_CCSD2_Supported is meaningful when CCSID2 is provided.

CUN4BIPR_Conversion_Supported - set by the service

Specifies whether the conversion between CCSIDs provided by CUN4BIPR_CCSD1 and CUN4BIPR_CCSD2 are supported, according the following values:

0 Conversion is not supported.

1 Conversion is supported.

CUN4BIPR_Conversion_Supported is meaningful when both CCSID1 and CCSID2 are provided.

CUN4BIPR_Gen_Flags_In - set by caller

CUN4BIPR_Get_Tech_Supp_fCCSID2_tCCSID1 -set by caller

Specifies whether techniques supported for CCSID2 to CCSID1 are returned at CUN4BIPR_Conv_Tech_fCCSID2_tCCSID1, according the following values:

0 Do not obtain techniques supported from CCSID2 to CCSID1. This is the default.

1 Obtain techniques supported from CCSID2 to CCSID1.

CUN4BIPR_Conv_Tech_fCCSID1_tCCSID2- set by the service

Specifies the conversion techniques supported for CCSID1 to CCSID2.
CUN4BIPR_Conv_Tech_fCCSID1_tCCSID2 is meaningful when
CUN4BIPR_Conversion_Supported is on.

CUN4BIPR_Conv_Tech_fCCSID2_tCCSID1- set by the service

Specifies the conversion techniques supported for CCSID2 to CCSID1.
CUN4BIPR_Conv_Tech_fCCSID2_tCCSID1 is meaningful when
CUN4BIPR_Conversion_Supported is on.

CUNBIPRM_DDA_Buf_Ptr - set by caller

Specifies the beginning address of an area of storage that the conversion
information services are using internally as dynamic data area.

Note: CUN4BIPR_DDA_Buf_Ptr must be double-word boundary.

CUNBIPRM_DDA_Buf_ALET - set by caller

Specifies the alet to be used if the dynamic data area addressed by
CUN4BIPR_DDA_Buf_Ptr resides in a different address or data space.

CUNBIPRM_DDA_Buf_Len - set by caller

Specifies the length in bytes of the dynamic data area addressed by
CUN4BIPR_DDA_Buf_Ptr. The required length is defined by constant
CUN4BIPR_DDA_Req that is provided in the interface definition file
CUN4BIID.

CUN4BIPR_RC_RS - set by the service

Specifies the return code and reason code.

CUN4BIPR_Return_Code - set by the service

Specifies the return code.

CUN4BIPR_Reason_Code - set by the service

Specifies the reason code.

CUN4BIPR_subCCSIDs_CCSID - set by the service

Specifies a subCCSIDs.

CUN4BIPR_subCCSIDs_Size - set by the service

Specifies the size character for the subCCSID.

CUN4BIPR_subCCSIDs_Description - set by the service

Specifies the description of the subCCSID.

Sample programs

Sample programs for conversion information service are provided in
SYS1.SAMPLIB:

31-bit samples

- CUNSISMC for C
- CUNSISMA for HLASM

64-bit samples

- CUN4SISC for C
- CUN4SISA for HLASM

Appendix A. MBCS conversions

This chapter describes information about the conversion of a MBCS CCSID.

Internal handling of MBCS conversions

Whenever a MBCS CCSID is specified for a conversion, z/OS support for Unicode decomposes the MBCS CCSID into its SBCS and DBCS parts. There are no MBCS tables provided for MBCS conversions.

As an example, if conversion from CCSID 939 to CCSID 13488 is specified, the MBCS CCSID 939 will be decomposed into the following sub CCSIDs:

- CCSID 1027 used for SBCS data in the input character stream
- CCSID 300 used for DBCS data in the input character stream

These CCSIDs are selected according to a predefined list.

In the example, the conversion service switches between the SBCS table and the DBCS table when a shift character is in the data stream.

Figure 1 on page 218 illustrates this method.

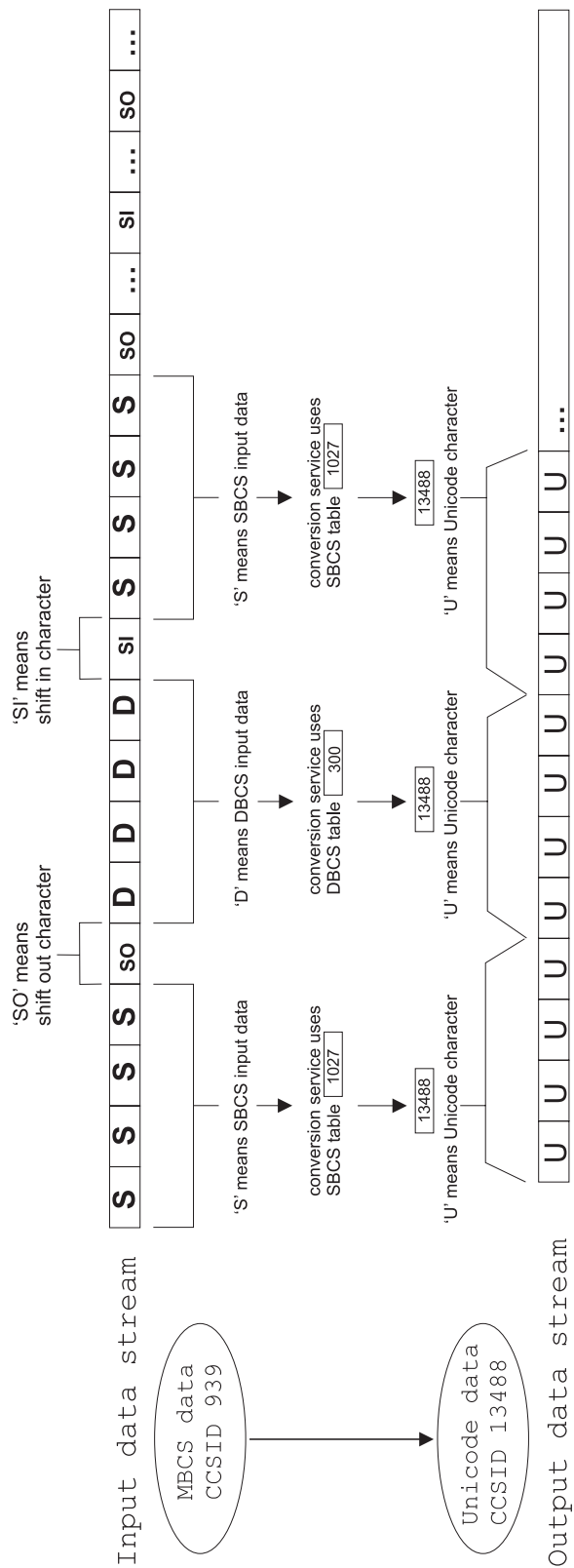


Figure 1. Conversion of MBCS data to Unicode characters

Shift characters in the input character stream specify if the subsequent data represents SBCS or DBCS characters. 'Shift out' character means that DBCS data will follow. 'Shift in' character means that SBCS data will follow. Thus, the

conversion service switches between the SBCS table and the DBCS table. (In Figure 1 on page 218 the 'shift out' character is indicated by **SO** and the 'shift in' character by **SI**).

The image generator selects one table that handles the SBCS part (CCSID 1027 to CCSID 13488) and another table which handles the DBCS part (CCSID 300 to CCSID 13488). The selection depends on the specified *technique-search-order* characters and the availability of the appropriate conversion tables.

For more information on how MBCS CCSIDs are composed, refer to Character Data Representation Architecture Reference .

MBCS CCSID decomposition

The following table shows all MBCS CCSIDs and how these CCSIDs can be decomposed into multiple CCSIDs (sub-CCSIDs) — SBCS and DBCS.

MBCS	Sub 1	Sub2	Sub3	Sub4	Sub5	Sub6
00930	00290	00300				
00931	08229	00300				
00932	00897	00301				
00933	00833	00834				
00934	00891	00926				
00935	00836	00837				
00936	00903	00928				
00937	28709	00835				
00938	00904	00927				
00939	01027	00300				
00942	01041	00301				
00943	13185	00941				
00944	01040	00926				
00946	01042	00928				
00948	01043	00927				
00949	01088	00951				
00950	01114	00947				
00954	00895	00952	09088	00953		
00956	00895	13240	00896	21433		
00957	00895	00955	00896	21433		
00958	00367	13240	00896	21433		
00959	00367	00955	00896	21433		
00964	00367	00960	00961			
00965	00367	05056	00963			
00966*	00367	05056	00963	05478		
00970	00367	00971				
01350	00367	05048	13184	05049		
01363	01126	01362				

MBCS CCSID decomposition

MBCS	Sub 1	Sub2	Sub3	Sub4	Sub5	Sub6
01364	13121	04930				
01370	05210	21427				
01371	01159	09027				
01373*	01114	01372				
01375	09444	01374				
01381	01115	01380				
01383	00367	01382				
01386	05210	01385				
01388	13124	04933				
01390	08482	16684				
01392	09444	09577	01391			
01394*	13185	01393				
01399	05123	16684				
05026	00290	04396				
05028	04993	00301				
05029	04929	00834				
05031	04932	00837				
05033	08229	00835				
05035	01027	04396				
05038	01041	08493				
05039	01041	01351				
05045	01088	05047				
05046	01114	05043				
05050	00895	00952	13184	09145		
05052	00895	13240	00896	21433		
05053	00895	00955	00896	21433		
05054	00367	13240	00896	21433		
05055	00367	00955	00896	21433		
05060*	00367	00960	05057			
05066*	00367	13259				
05459*	05222	05458				
05460*	00833	17218				
05471	09444	05470				
05473	28709	05472				
05477	05211	01380				
05479	00367	09574				
05482*	01114	05481				
05484*	00836	09029				
05486*	12578	20780				
05488	09444	09577	05487			
05495*	09219	20780				

MBCS CCSID decomposition

MBCS	Sub 1	Sub2	Sub3	Sub4	Sub5	Sub6
09122	04386	00300				
09124	09089	00301				
09125	09025	09026				
09127	09028	00837				
09131	01027	08493				
09135*	00897	09133				
09142	01114	09139				
09146	00895	00952	13184	00953		
09148*	00367	00895	00955	17336		
09555*	05222	09554				
09575	00367	05478				
09580	00836	13125				
13218	04386	04396				
13219	08229	04396				
13231*	00897	17325				
13238	01114	13235				
13242	00895	05048	13184	05049		
13651*	05222	13650				
13671*	00367	05478				
13676	00836	17221				
17314	00290	12588				
17317*	09025	13122				
17354	00367	09163				
21450	00367	05067				
25508	25473	24877				
25510	25467	25502				
25512	25479	25504				
25514	25480	25503				
25518	25617	24877				
25520	25616	25502				
25522	25618	25504				
25524	25619	25503				
25525	25664	25527				
25546	00367	09163				
29614	29713	24877				
29616	29712	25502				
29618	29714	25504				
29620	29715	25503				
29621	29760	25527				
33698	33058	04396				
33699	32805	04396				

MBCS CCSID decomposition

MBCS	Sub 1	Sub2	Sub3	Sub4	Sub5	Sub6
33700	33665	24877				
33717	25664	29623				
33722	00895	00952	09088	09145		
37796	37761	24877				
37813	29760	29623				
57345*	00367	00895	00896		05049	00955

Note: The MBCS CCSIDs that have an asterisk (*) in column one are not supported by z/OS support for Unicode.

MBCS CCSIDs compatible with iconv

The following is a list of MBCS CCSID tables that were changed to provide compatibility with the Language Environment iconv() function.

These CCSIDs can be selected by using the technique character "L" when calling the service and when defining conversions for the image generator.

00930
00932
00939
00958
00959
05054
05055

If you are looking for iconv() compatible SBCS and DBCS tables, any "EBCDIC Conversion Table" on page 248, "ASCII Conversion Table" on page 326, or "Unicode Conversion Table" on page 419 described in Appendix B, "Conversion Tables Supplied with z/OS Unicode," on page 223 that support technique L can be used. Technique L is described in "Creating a conversion image" on page 9.

C-variant MBCS CCSIDs compatible with iconv()

The following is a list of MBCS CCSID tables that were changed to provide compatibility with C-variants when using the Language Environment iconv() function.

These CCSIDs can be selected by using the technique character "M" when calling the service and when defining conversions for the image generator.

00932 corresponds to IBM-932C
00942 corresponds to IBM-942C
00943 corresponds to IBM-943C
00949 corresponds to IBM-949C
01363 corresponds to IBM-1363C
33722 corresponds to IBM-eucJC

Appendix B. Conversion Tables Supplied with z/OS Unicode

The following conversion tables are supplied:

- Description of CCSIDs
- EBCDIC Conversion Table
- ASCII Conversion Table
- Unicode Conversion Table
- Case Conversion Table
- Normalization Table
- Collation Table
- Stringprep Table

Description of CCSIDs

The following table describes the CCSIDs supported by z/OS Unicode.

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
00037	EBCDIC, SBCS	USA, CANADA, BRAZIL, and COMMON EUROPE	AA
00256	EBCDIC, SBCS	NETHERLAND	AJ
00259	EBCDIC, SBCS	SYMBOLS SET 7	AP
00273	EBCDIC, SBCS	AUSTRIA and GERMANY	AV
00274	EBCDIC, SBCS	BELGIUM	AX
00275	EBCDIC, SBCS	BRAZIL	AZ
00277	EBCDIC, SBCS	DENMARK, NORWAY	A2
00278	EBCDIC, SBCS	FINLAND, SWEDEN	A4
00280	EBCDIC, SBCS	ITALIAN	A6
00281	EBCDIC, SBCS	JAPAN	A8
00282	EBCDIC, SBCS	PORTUGAL	A9
00284	EBCDIC, SBCS	SPANISH	BB
00285	EBCDIC, SBCS	UNITED KINGDOM	BE
00286	EBCDIC, SBCS	AUSTRIA and GERMANY 3279	BG
00290	EBCDIC, SBCS	JAPANESE	BH
00293	EBCDIC, SBCS	APL (A Programming Language) USA	BL
00297	EBCDIC, SBCS	FRENCH	BN
00300	EBCDIC, DBCS	JAPAN	BQ
00301	ASCII, DBCS	JAPAN	BV
00310*	EBCDIC, SBCS	APL (A Programming Language) /TN	BZ

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
00367	ASCII, SBCS	USA, ANSI X3.4 ASCII STANDAR	B0
00420	EBCDIC, SBCS	ARABIC	B1
00421	EBCDIC, SBCS	MAGHREB/FRENCH	B6
00423	EBCDIC, SBCS	GREEK	B8
00424	EBCDIC, SBCS	HEBREW	CA
00425	EBCDIC, SBCS	ARABIC/LATIN	SR
00437	ASCII, SBCS	USA	CE
00500	EBCDIC, SBCS	INTERNATIONAL	CR
00720	ASCII, SBCS	MICROSOFT-DOS ARABIC	C5
00737	ASCII, SBCS	MICROSOFT-DOS GREEK	C6
00775	ASCII, SBCS	MICROSOFT-DOS BALTIC	C8
00803	EBCDIC, SBCS	HEBREW	DA
00806	ASCII, SBCS	PC-ISCII-91	DC
00807*	ASCII, SBCS	ISCII	DE
00808	ASCII, SBCS	CYRILLIC	D5
00813	ASCII, SBCS	GREEK/LATIN	DF
00819	ASCII, SBCS	ISO 8859-1	DH
00833	EBCDIC, SBCS	KOREAN	DI
00834	EBCDIC, DBCS	KOREAN	DM
00835	EBCDIC, DBCS	TRADITIONAL CHINESE (T-CH)	DR
00836	EBCDIC, SBCS	SIMPLIFIED CHINESE (S-CH)	DU
00837	EBCDIC, DBCS	SIMPLIFIED CHINESE (S-CH)	DY
00838	EBCDIC, SBCS	THAILAND	D1
00839*	EBCDIC, DBCS	THAI	D4
00848	ASCII, SBCS	UKRAINE	D7
00849	ASCII, SBCS	BELARUS	D9
00850	ASCII, SBCS	LATIN-1	EB
00851	ASCII, SBCS	GREEK	EG
00852	ASCII, SBCS	LATIN-2	EL
00853	ASCII, SBCS	TURKISH	ES
00855	ASCII, SBCS	CYRILLIC	EX
00856	ASCII, SBCS	HEBREW	E4
00857	ASCII, SBCS	TURKISH	FC
00858	ASCII, SBCS	LATIN-1E	FI
00859	ASCII, SBCS	LATIN-9	FK

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
00860	ASCII, SBCS	PORTUGUESE	FM
00861	ASCII, SBCS	ICELAND	FP
00862	ASCII, SBCS	HEBREW	FS
00863	ASCII, SBCS	CANADA	FV
00864	ASCII, SBCS	ARABIC	FY
00865	ASCII, SBCS	DENMARK, NORWAY	GA
00866	ASCII, SBCS	CYRILLIC	GD
00867	ASCII, SBCS	HEBREW	GF
00868	ASCII, SBCS	URDU	GH
00869	ASCII, SBCS	GREEK	GP
00870	EBCDIC, SBCS	LATIN-2	GW
00871	EBCDIC, SBCS	ICELAND	GY
00872	ASCII, SBCS	CYRILLIC	G0
00874	ASCII, SBCS	THAI PC-DATA	G3
00875	EBCDIC, SBCS	GREEK	G8
00878	ASCII, SBCS	KOI8-R CYRILLIC	HA
00880	EBCDIC, SBCS	CYRILLIC	HB
00891	ASCII, SBCS	KOREA	HD
00892*	EBCDIC, SBCS	OCR (OPTICAL CHARACTER RECOGNITION) -A	HF
00893*	EBCDIC, SBCS	OCR (OPTICAL CHARACTER RECOGNITION) -B	HG
00895	ASCII, SBCS	JAPAN 7-BIT LATIN	HH
00896	ASCII, SBCS	JAPAN 7-BIT KATAKANA	HI
00897	ASCII, SBCS	JAPAN	HK
00899	ASCII, SBCS	SYMBOLS - PC	HR
00901	ASCII, SBCS	BALTIC ISO-8	HS
00902	ASCII, SBCS	ESTONIA ISO-8	HU
00903	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH)	HW
00904	ASCII, SBCS	TRADITIONAL CHINESE (T-CH)	HY
00905	EBCDIC, SBCS	TURKEY	H0
00912	ASCII, SBCS	LATIN 2, ISO 8859-2	H1
00913	ASCII, SBCS	ISO LATIN 3, ISO 8859-3	SZ
00914	ASCII, SBCS	LATIN 4, ISO 8859-4	H3
00915	ASCII, SBCS	CYRILLIC, 8-BIT, ISO 8859-5	H4

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
00916	ASCII, SBCS	ISO 8859-8:HEBREW (string type 5)	H6
00918	EBCDIC, SBCS	URDU	H8
00920	ASCII, SBCS	ISO 8859-9 LATIN 5	IA
00921	ASCII, SBCS	BALTIC, 8-BIT(ISO 8859-13)	IB
00922	ASCII, SBCS	ESTONIA ISO-8	ID
00923	ASCII, SBCS	ISO 8859-15	IF
00924	EBCDIC, SBCS	LATIN 9	IG
00926	ASCII, DBCS	KOREA	IH
00927	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)	IJ
00928	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH)	IM
00929*	ASCII, DBCS	THAI	IO
00930	EBCDIC, MBCS	JAPANESE KATAKANA- KANJI	IQ
00931	EBCDIC, MBCS	JAPANESE LATIN-KANJI	IW
00932	ASCII, MBCS	JAPAN	IZ
00933	EBCDIC, MBCS	KOREAN	I5
00934	ASCII, MBCS	KOREAN	JA
00935	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH)	JC
00936	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH)	JG
00937	EBCDIC, MBCS	TRADITIONAL CHINESE (T-CH)	JI
00938	ASCII, MBCS	TRADITIONAL CHINESE (T-CH)	JK
00939	EBCDIC, MBCS	JAPANESE LATIN - KANJI	JM
00941	ASCII, DBCS	JAPANESE PC FOR OPEN ENVIRONMENT	JP
00942	ASCII, MBCS	JAPAN	JU
00943	ASCII, MBCS	JAPAN OPEN	JY
00944	ASCII, MBCS	KOREA	J3
00946	ASCII, MBCS	SIMPLIFIED CHINESE (S- CH)	J6
00947	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG-5	J9
00948	ASCII, MBCS	TRADITIONAL CHINESE (T-CH)	KF

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
00949	ASCII, MBCS	KOREA KS	KI
00950	ASCII, MBCS	TRADITIONAL CHINESE (T-CH)	KO
00951	ASCII, DBCS	IBM KS	KS
00952	ASCII, DBCS	JAPANESE EUC	KW
00953	ASCII, DBCS	JAPANESE EUC	KY
00954	ASCII, MBCS	JAPANESE EUC	K1
00955	ASCII, DBCS	JAPANESE TCP	K6
00956	ASCII, MBCS	JAPANESE TCP	K7
00957	ASCII, MBCS	JAPANESE TCP	K9
00958	ASCII, MBCS	JAPANESE TCP	LB
00959	ASCII, MBCS	JAPANESE TCP	LD
00960	ASCII, DBCS	TRADITIONAL CHINESE (T-CH) EUC	LF
00961	ASCII, TBCS	TRADITIONAL CHINESE (T-CH) EUC	LG
00963	ASCII, DBCS	TRADITIONAL CHINESE (T-CH) TCP	LI
00964	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) EUC	LJ
00965	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) TCP	LL
00966	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) TCP	LN
00970	ASCII, MBCS	KOREAN EUC	LO
00971	ASCII, DBCS	KOREAN EUC	LT
01002	EBCDIC, SBCS	DCF RELEASE 2 COMPATIBILITY	LV
01004	ASCII, SBCS	LATIN-1	LW
01006	ASCII, SBCS	URDU ISO- 8	LZ
01008	ASCII, SBCS	ARABIC ISO/ASCII	L0
01009	ASCII, SBCS	ISO-7: IRV (International reference version)	L2
01010	ASCII, SBCS	ISO-7: FRENCH	L3
01011	ASCII, SBCS	ISO-7: GERMANY	L4
01012	ASCII, SBCS	ISO-7: ITALY	L5
01013	ASCII, SBCS	ISO-7: UNITED KINGDOM	L6

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
01014	ASCII, SBCS	ISO-7: SPAIN	L7
01015	ASCII, SBCS	ISO-7: PORTUGAL	L8
01016	ASCII, SBCS	ISO-7: NORWAY	L9
01017	ASCII, SBCS	ISO-7: DENMARK	MA
01018	ASCII, SBCS	ISO-7: FINLAND and SWEDEN	MB
01019	ASCII, SBCS	ISO-7: BELGIUM and NETHERLANDS	MC
01020	ASCII, SBCS	ISO-7: CANADA	MD
01021	ASCII, SBCS	ISO-7: SWITZERLAND VARIANT	ME
01023	ASCII, SBCS	ISO-7: SPAIN	MF
01025	EBCDIC, SBCS	CYRILLIC MULTILINGUAL	MG
01026	EBCDIC, SBCS	TURKEY LATIN-5	MH
01027	EBCDIC, SBCS	JAPAN LATIN	MI
01040	ASCII, SBCS	KOREA	MK
01041	ASCII, SBCS	JAPAN	MN
01042	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH)	MR
01043	ASCII, SBCS	TRADITIONAL CHINESE (T-CH)	MU
01046	ASCII, SBCS	ARABIC - PC	MX
01047	EBCDIC, SBCS	LATIN 1/ OPEN SYSTEM	M0
01051	ASCII, SBCS	HP EMULATION	M2
01088	ASCII, SBCS	KOREA KS	M3
01089	ASCII, SBCS	ARABIC ISO 8859-6	M6
01097	EBCDIC, SBCS	FARSI	M7
01098	ASCII, SBCS	FARSI PC	M8
01100	ASCII, SBCS	MULTI EMULATION	M9
01101	ASCII, SBCS	BRITISH ISO-7 NRC SET	NA
01102	ASCII, SBCS	DUTCH ISO-7 NRC SET	NB
01103	ASCII, SBCS	FINNISH ISO-7 NRC SET	NC
01104	ASCII, SBCS	FRENCH ISO-7 NRC SET	ND
01105	ASCII, SBCS	NOR/DAN ISO-7 NRC SET	NE
01106	ASCII, SBCS	SWEDISH ISO-7 NRC SET	NF

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
01107	ASCII, SBCS	NOR/DAN ISO-7 NRC SET	NG
01112	EBCDIC, SBCS	BALTIC	NH
01114	ASCII, SBCS	TRADITIONAL CHINESE (T-CH)	NI
01115	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) GB	NM
01122	EBCDIC, SBCS	ESTONIA	NP
01123	EBCDIC, SBCS	UKRAINE	NQ
01124	ASCII, SBCS	UKRAINE ISO-8	NR
01125	ASCII, SBCS	UKRAINE	NS
01126	ASCII, SBCS	KOREAN MS-WIN	NT
01127*	ASCII, SBCS	ARABIC / FRENCHG	NW
01129	ASCII, SBCS	VIETNAMESE ISO-8	NY
01130	EBCDIC, SBCS	VIETNAMESE	NZ
01131	ASCII, SBCS	BELARUS	N0
01132	EBCDIC, SBCS	LAO	N1
01133	ASCII, SBCS	LAO ISO-8	N2
01137	EBCDIC, SBCS	DEVANAGARI	N3
01139*	ASCII, SBCS	JAPAN 8-BIT	N4
01140	EBCDIC, SBCS	COMMON EUROPE ECECP	N5
01141	EBCDIC, SBCS	AUSTRIA and GERMANY ECECP	N6
01142	EBCDIC, SBCS	DENMARK, NORWAY ECECP	N7
01143	EBCDIC, SBCS	FINLAND, SWEDEN ECECP	N8
01144	EBCDIC, SBCS	ITALIAN ECECP	N9
01145	EBCDIC, SBCS	SPANISH ECECP	OA
01146	EBCDIC, SBCS	UNITED KINGDOM ECECP	OB
01147	EBCDIC, SBCS	FRENCH ECECP	OC
01148	EBCDIC, SBCS	INTERNATIONAL ECECP	OD
01149	EBCDIC, SBCS	ICELAND ECECP	OE
01153	EBCDIC, SBCS	LATIN-2	OF
01154	EBCDIC, SBCS	CYRILLIC	OG
01155	EBCDIC, SBCS	TURKEY LATIN-5	OH
01156	EBCDIC, SBCS	BALTIC	OI
01157	EBCDIC, SBCS	ESTONIA	OJ
01158	EBCDIC, SBCS	UKRAINE	OK

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
01159	EBCDIC, SBCS	TRADITIONAL CHINESE (T-CH)	OL
01160	EBCDIC, SBCS	THAI	OM
01161	ASCII, SBCS	THAI	ON
01162	ASCII, SBCS	THAI WINDOWS	OO
01163	ASCII, SBCS	VIETNAMESE ISO8	OP
01164	EBCDIC, SBCS	VIETNAMESE	OQ
01165	EBCDIC, SBCS	LATIN-2 OPEN SYSTEM	SV
01166	EBCDIC, SBCS	CYRILLIC MULTILINGUAL - Kazakhstan	TN
01167	ASCII, SBCS	BELARUSIAN / UKRAINIAN KOI8-RU	TO
01168	ASCII, SBCS	UKRAINIAN KOI8-U	TP
01169*	ASCII, SBCS	WINDOWS, CYRILLIC ASIAN	TS
01200	UCS-2, DBCS	UTF-16	PF
01201*	UCS-2, DBCS	UTF-16	PI
01202	UTF-16, DBCS	UTF-16 LE with IBM PUA	T7
01208	UCS-2, UTF-8	UTF-8	PK
01209*	UCS-2, UTF-8	UTF-8	PM
01250	ASCII, SBCS	MS-WIN LATIN-2	PO
01251	ASCII, SBCS	MS-WIN CYRILLIC	PQ
01252	ASCII, SBCS	MS-WIN LATIN-1	PS
01253	ASCII, SBCS	MS-WIN GREEK	PU
01254	ASCII, SBCS	MS-WIN TURKEY	PW
01255	ASCII, SBCS	MS-WIN HEBREW	PY
01256	ASCII, SBCS	MS-WIN ARABIC	P0
01257	ASCII, SBCS	MS-WIN BALTIC	P2
01258	ASCII, SBCS	MS-WIN VIETNAM	P4
01275	ASCII, SBCS	APPLE LATIN- 1	P6
01276	ASCII, SBCS	ADOBE STANDARD	P7
01277	ASCII, SBCS	ADOBE LATIN- 1	P8
01280	ASCII, SBCS	APPLE GREEK	QA
01281	ASCII, SBCS	APPLE TURKEY	QB
01282	ASCII, SBCS	APPLE LATIN2	QC
01283	ASCII, SBCS	APPLE CYRILLIC	QD
01284	ASCII, SBCS	APPLE CROATIAN	QE
01285	ASCII, SBCS	APPLE ROMANIAN	QF
01286*	ASCII, SBCS	APPLE ICELANDIC	QG

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
01287	ASCII, SBCS	DEC (DIGITAL EQUIPMENT CORPORATION) GREEK 8-Bit	SX
01288	ASCII, SBCS	DEC (DIGITAL EQUIPMENT CORPORATION) TURKISH 8-Bit	SY
01350	ASCII, MBCS	JIS JAPANESE EUC	QH
01351	ASCII, DBCS	JAPAN OPEN	QI
01362	ASCII, DBCS	KOREAN MS-WIN	QJ
01363	ASCII, MBCS	KOREAN MS- WIN	QN
01364	EBCDIC, MBCS	KOREAN	QR
01370	ASCII, MBCS	TRADITIONAL CHINESE (T-CH)	QT
01371	EBCDIC, MBCS	TRADITIONAL CHINESE (T-CH)	QU
01374	ASCII, DBCS	IBM BIG-5 EXTENSION FOR HKSCS	TZ
01375	ASCII, MBCS	IBM BIG-5 EXTENSION FOR HKSCS	TY
01380	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) GB	QV
01381	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) GB	QY
01382	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) EUC	Q0
01383	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) EUC to GB 2312	Q2
01385	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) GBK	Q6
01386	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) GBK	Q8
01388	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH)	RA
01390	EBCDIC, MBCS	JAPAN	RC
01391	ASCII, QBCS	SIMPLIFIED CHINESE (S-CH)-growing for GB18030	TF

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
01392	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH)-growing for GB18030	TG
01393*	ASCII, DBCS	JAPANESE JISx0213 (based on JISX0213)	TV
01399	EBCDIC, MBCS	JAPAN	RD
04133	EBCDIC, SBCS	USA	AB
04369	EBCDIC, SBCS	AUSTRIA and GERMANY	AW
04370	EBCDIC, SBCS	BELGIUM	AY
04371	EBCDIC, SBCS	BRAZIL	A0
04372*	EBCDIC, SBCS	CANADA	A1
04373	EBCDIC, SBCS	DENMARK, NORWAY	A3
04374	EBCDIC, SBCS	FINLAND, SWEDEN	A5
04376	EBCDIC, SBCS	ITALY	A7
04378	EBCDIC, SBCS	PORTUGAL	BA
04380	EBCDIC, SBCS	LATIN	BC
04381	EBCDIC, SBCS	UNITED KINGDOM	BF
04386	EBCDIC, SBCS	JAPAN	BI
04389*	EBCDIC, SBCS	APL (A Programming Language)	BM
04393	EBCDIC, SBCS	FRANCE	BO
04396	EBCDIC, DBCS	JAPAN	BR
04397	ASCII, DBCS	JAPAN	BW
04516	EBCDIC, SBCS	ARABIC	B2
04517	EBCDIC, SBCS	MAGHREB/FRENCH	B7
04519	EBCDIC, SBCS	GREEK 3174	B9
04520	EBCDIC, SBCS	HEBREW	CB
04533	ASCII, SBCS	SWISS	CF
04596	EBCDIC, SBCS	LATIN AMERICA	CS
04899	EBCDIC, SBCS	HEBREW	DB
04902*	ASCII, SBCS	PC-ISCII	DD
04904	ASCII, SBCS	CYRILLIC (with MS controls)	OS
04909	ASCII, SBCS	GREEK/LATIN	DG
04929	EBCDIC, SBCS	KOREA	DJ
04930	EBCDIC, DBCS	KOREAN	DN
04931	EBCDIC, DBCS	TRADITIONAL CHINESE (T-CH)	DS
04932	EBCDIC, SBCS	SIMPLIFIED CHINESE (S-CH)	DV

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
04933	EBCDIC, DBCS	SIMPLIFIED CHINESE (S-CH)	DZ
04934	EBCDIC, SBCS	THAI	D2
04944	ASCII, SBCS	UKRAINE (with MS controls)	OT
04945	ASCII, SBCS	BELARUS (with MS controls)	OU
04946	ASCII, SBCS	LATIN-1	EC
04947	ASCII, SBCS	GREEK	EH
04948	ASCII, SBCS	LATIN-2	EM
04949	ASCII, SBCS	TURKEY	ET
04951	ASCII, SBCS	CYRILLIC	EY
04952	ASCII, SBCS	HEBREW	E5
04953	ASCII, SBCS	TURKEY	FD
04954	ASCII, SBCS	LATIN-1E (with MS controls)	OY
04955	ASCII, SBCS	LATIN-9 (with MS controls)	OZ
04956	ASCII, SBCS	PORTUGESE (with MS controls)	O0
04957	ASCII, SBCS	ICELAND (with MS controls)	O1
04958	ASCII, SBCS	HEBREW (with MS controls)	O2
04959	ASCII, SBCS	CANADA (with MS controls)	O3
04960	ASCII, SBCS	ARABIC	FZ
04961	ASCII, SBCS	DENMARK, NORWAY	O4
04962	ASCII, SBCS	CYRILLIC (with MS controls)	O5
04963	ASCII, SBCS	HEBREW (with MS controls)	O6
04964	ASCII, SBCS	URDU	GI
04965	ASCII, SBCS	GREEK	GQ
04966	EBCDIC, SBCS	ROECE LATIN-2	GX
04967	EBCDIC, SBCS	ICELAND	GZ
04968*	ASCII, SBCS	CYRILLIC	G1
04970	ASCII, SBCS	THAI	G4
04971	EBCDIC, SBCS	GREEK	G9
04976	EBCDIC, SBCS	CYRILLIC	HC
04992	ASCII, SBCS	JAPANESE TCP-2022	HJ
04993	ASCII, SBCS	JAPAN	HL

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
05012	ASCII, SBCS	ISO 8859-8	H7
05014	EBCDIC, SBCS	URDU	H9
05023	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)	IK
05026	EBCDIC, MBCS	JAPAN	IR
05028	ASCII, MBCS	JAPAN	I0
05029	EBCDIC, MBCS	KOREA	I6
05031	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH)	JD
05033	EBCDIC, MBCS	TRADITIONAL CHINESE (T-CH)	JJ
05035	EBCDIC, MBCS	JAPAN MIX	JN
05037*	ASCII, DBCS	JAPAN OPEN	JQ
05038	ASCII, MBCS	JAPAN HP15-J (Defined by Hewlett Packard)	JV
05039	ASCII, MBCS	JAPAN OPEN	JZ
05043	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG-5	KA
05045	ASCII, MBCS	KOREA KS	KJ
05046	ASCII, MBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG-5	KP
05047	ASCII, DBCS	KOREA KS PC DATA	KT
05048	ASCII, DBCS	JAPANESE EUC	KX
05049	ASCII, DBCS	JAPANESE EUC	KZ
05050	ASCII, MBCS	JAPANESE EUC	K2
05052	ASCII, MBCS	JAPANESE TCP	K8
05053	ASCII, MBCS	JAPANESE TCP	LA
05054	ASCII, MBCS	JAPANESE TCP	LC
05055	ASCII, MBCS	JAPANESE TCP	LE
05056	ASCII, DBCS	TRADITIONAL CHINESE (T-CH) TCP-2022	SS
05057*	ASCII, TBCS	TRADITIONAL CHINESE (T-CH) EUC	LH
05060*	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) EUC	LK
05061*	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) TCP	LM
05066*	ASCII, MBCS	KOREAN EUC	LP

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
05067	ASCII, DBCS	KOREAN EUC	LU
05100	ASCII, SBCS	LATIN-1	LX
05104	ASCII, SBCS	ARABIC ISO/ASCII	L1
05123	EBCDIC, SBCS	JAPAN LATIN	MJ
05137	ASCII, SBCS	JAPAN	MO
05142	ASCII, SBCS	ARABIC - PC	MY
05143	EBCDIC, SBCS	LATIN OPEN SYS	M1
05210	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) SB	NJ
05211	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) GB	NN
05222*	ASCII, SBCS	Korean MS-WIN	NU
05297*	UCS-2, DBCS	UCS-2	PJ
05304*	UCS-2, UTF-8	UTF-8	PL
05305*	UCS-2, UTF-8	UTF-8	PN
05346	ASCII, SBCS	MS-WIN LATIN-2	PP
05347	ASCII, SBCS	MS-WIN CYRILLIC	PR
05348	ASCII, SBCS	MS-WIN LATIN-1	PT
05349	ASCII, SBCS	MS-WIN GREEK	PV
05350	ASCII, SBCS	MS-WIN TURKEY	PX
05351	ASCII, SBCS	MS-WIN HEBREW	PZ
05352	ASCII, SBCS	MS-WIN ARABIC	P1
05353	ASCII, SBCS	MS-WIN BALTIC	P3
05354	ASCII, SBCS	MS-WIN VIETNAM	P5
05458*	ASCII, DBCS	KOREAN MS- WIN	QK
05459*	ASCII, MBCS	KOREAN MS-WIN	QO
05460*	EBCDIC, MBCS	KOREAN	QS
05470	ASCII, DBCS	Big-5 extension for HKSCS 2001	T2
05471	EBCDIC, MBCS	IBM BIG-5 EXTENSION FOR HKSCS	T1
05472	EBCDIC, DBCS	Host HKSCS-2001	T4
05473	EBCDIC, MBCS	T-Chinese Mixed Host for HKSCS	T3
05476	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) GB	QW
05477	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) GB	QZ
05478	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) EUC	Q1

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
05479	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) EUC	Q3
05481*	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) GBK	Q7
05482*	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) GBK	Q9
05484*	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH)	RB
05487	ASCII, QBCS	SIMPLIFIED CHINESE (S-CH)- for GB 18030	TC
05488	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) - GB18030	TB
08229	EBCDIC, SBCS	INTERNATIONAL	AC
08448	EBCDIC, SBCS	INTERNATIONAL	AK
08476*	EBCDIC, SBCS	SPAIN	BD
08482	EBCDIC, SBCS	JAPAN	BJ
08489*	EBCDIC, SBCS	FRANCE	BP
08492	EBCDIC, DBCS	JAPAN	BS
08493	ASCII, DBCS	JAPAN HP15-J (Defined by Hewlett Packard)	BX
08612	EBCDIC, SBCS	ARABIC	B3
08616*	EBCDIC, SBCS	HEBREW	CC
08629	ASCII, SBCS	AUSTRIA and GERMANY PC-DATA	CG
08692	EBCDIC, SBCS	AUSTRIA and GERMANY	CT
09025	EBCDIC, SBCS	KOREA	DK
09026	EBCDIC, DBCS	KOREA	DO
09027	EBCDIC, DBCS	TRADITIONAL CHINESE (T-CH)	DT
09028	EBCDIC, SBCS	SIMPLIFIED CHINESE (S-CH)	DW
09029*	EBCDIC, DBCS	SIMPLIFIED CHINESE (S-CH)	D0
09030	EBCDIC, SBCS	THAI	D3
09042	ASCII, SBCS	LATIN-1 (with MS controls)	OV
09044	ASCII, SBCS	LATIN-2	EN
09047	ASCII, SBCS	CYRILLIC	EZ
09048	ASCII, SBCS	HEBREW	E6

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
09049	ASCII, SBCS	TURKISH	FE
09056	ASCII, SBCS	ARABIC	F0
09060	ASCII, SBCS	URDU	GJ
09061	ASCII, SBCS	GREEK	GR
09064	ASCII, SBCS	CYRILLIC (with MS controls)	O8
09066	ASCII, SBCS	THAI	G5
09088	ASCII, SBCS	Japanese EUC, G2-JIS	S0
09089	ASCII, SBCS	JAPAN	HM
09122	EBCDIC, MBCS	JAPAN	IS
09124	ASCII, MBCS	JAPAN	I1
09125	EBCDIC, MBCS	KOREA	I7
09127	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH)	JE
09131	EBCDIC, MBCS	JAPAN	JO
09133*	ASCII, DBCS	JAPAN PC	JR
09135*	ASCII, MBCS	JAPAN PC	J0
09139	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG-5	KB
09142	ASCII, MBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG -5	KQ
09144	ASCII, DBCS	JAPANESE TCP-2022, G1	S1
09145	ASCII, DBCS	JAPANESE EUC	K0
09146	ASCII, MBCS	JAPANESE EUC	K3
09148*	ASCII, MBCS	JAPANESE TCP	SU
09163	ASCII, DBCS	KOREAN EUC, G1	S2
09219*	EBCDIC, SBCS	JAPAN LATIN	TR
09238	ASCII, SBCS	ARABIC - PC	MZ
09306	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) (with MS controls)	PA
09444	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) -part of GB 18030	TE
09447	ASCII, SBCS	MS-WIN HEBREW-2001	TM
09448	ASCII, SBCS	MS-WIN ARABIC-2001	TT
09449	ASCII, SBCS	MS-WIN BALTIC-2001	TU
09554*	ASCII, DBCS	KOREAN MS-WIN	QL

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
09555*	ASCII, MBCS	KOREAN MS-WIN	QP
09572	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) GB	QX
09574	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) EUC	S9
09575	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) TCP	Q4
09577	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) GBK	TD
09580	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH) Host for GBK	TI
12325*	EBCDIC, SBCS	CANADA	AD
12544	EBCDIC, SBCS	FRANCE	AL
12578*	EBCDIC, SBCS	JAPAN	SW
12588	EBCDIC, DBCS	JAPAN	BT
12708*	EBCDIC, SBCS	ARABIC	B4
12712	EBCDIC, SBCS	HEBREW	CD
12725	ASCII, SBCS	FRANCE	CH
12788	EBCDIC, SBCS	ITALY	CU
13121	EBCDIC, SBCS	KOREA	DL
13122*	EBCDIC, DBCS	KOREAN	DP
13124	EBCDIC, SBCS	SIMPLIFIED CHINESE (S-CH)	DX
13125	EBCDIC, DBCS	SIMPLIFIED CHINESE (S-CH)-Host-for GBK	TJ
13140	ASCII, SBCS	LATIN-2 (with MS controls)	PB
13143	ASCII, SBCS	CYRILLIC (with MS controls)	OW
13144*	ASCII, SBCS	HEBREW	E7
13145	ASCII, SBCS	TURKISH (with MS controls)	PC
13152	ASCII, SBCS	ARABIC	F1
13156	ASCII, SBCS	URDU (with MS controls)	O7
13157	ASCII, SBCS	GREEK (with MS controls)	PD
13162	ASCII, SBCS	THAI (with MS controls)	O9

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
13184	ASCII, SBCS	JAPAN 7-BIT KATAKANA	S5
13185	ASCII, SBCS	JAPAN	HN
13218	EBCDIC, MBCS	JAPAN	IT
13219	EBCDIC, MBCS	JAPAN	IX
13221	EBCDIC, MBCS	KOREA	I8
13223	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH)	JF
13229*	ASCII, DBCS	JAPAN PC	JS
13231*	ASCII, MBCS	JAPAN PC	J2
13235	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG -5	KC
13238	ASCII, MBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG-5	KR
13240	ASCII, DBCS	JAPANESE TCP-2022	S6
13241	ASCII, DBCS	Japanese TCP-2022	S3
13242	ASCII, MBCS	JAPANESE EUC	K4
13259*	ASCII, DBCS	KOREAN EUC	S8
13488	UCS-2, DBCS	UCS-2 version 3.0	PG
13650*	ASCII, DBCS	KOREAN MS-WIN	QM
13651*	ASCII, MBCS	KOREAN MS-WIN	QQ
13671	ASCII, MBCS	SIMPLIFIED CHINESE (S- CH) TCP	Q5
13676	EBCDIC, MBCS	SIMPLIFIED CHINESE (S-CH)-Host for GBK	TK
16421	EBCDIC, SBCS	CANADA	AE
16684	EBCDIC, DBCS	JAPAN	BU
16804	EBCDIC, SBCS	ARABIC	B5
16821	ASCII, SBCS	ITALY	CI
16884	EBCDIC, SBCS	FINLAND, SWEDEN	CV
17218*	EBCDIC, DBCS	KOREAN	DQ
17221	EBCDIC, DBCS	SIMPLIFIED CHINESE (S-CH)-Host for GBK	TL
17240	ASCII, SBCS	HEBREW (with MS controls)	OX
17248	ASCII, SBCS	ARABIC	F2
17314	EBCDIC, MBCS	JAPAN	IU
17317*	EBCDIC, MBCS	KOREA	I9

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
17325*	ASCII, DBCS	JAPAN PC	JT
17331	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG-5	KD
17336*	ASCII, DBCS	JAPANESE TCP-2022	TA
17337	ASCII, DBCS	Japanese TCP-2022 G3-JIS	S4
17354	ASCII, MBCS	KOREAN TCP	LQ
17584	UCS-2, DBCS	UCS-2 (version 3.1)	PH
20517	EBCDIC, SBCS	PORTUGAL	AF
20780	EBCDIC, DBCS	JAPAN	TQ
20917	ASCII, SBCS	UNITED KINGDOM PC- DATA	CJ
20980	EBCDIC, SBCS	DENMARK, NORWAY	CW
21314	EBCDIC, DBCS	KOREAN	TW
21317	EBCDIC, DBCS	SIMPLIFIED CHINESE (S-CH)	TX
21344	ASCII, SBCS	ARABIC (with MS controls)	PE
21427	ASCII, DBCS	TRADITIONAL CHINESE (T-CH)-IBM BIG-5	KE
21433	ASCII, DBCS	JAPANESE EUC	S7
21450	ASCII, MBCS	KOREAN TCP	LR
21680	UCS-2, DBCS	UCS-2 (version 4.0)	TH
24613	EBCDIC, SBCS	INTERNATIONAL	AG
24877	ASCII, DBCS	JAPAN PC-DISPLAY	BY
25013	ASCII, SBCS	USA PC-DISPLAY	CK
25076	EBCDIC, SBCS	DENMARK, NORWAY	CX
25313*	ASCII, SBCS	MICROSOFT-DOS GREEK	C7
25351*	ASCII, SBCS	MICROSOFT-DOS BALTIC	C9
25384*	ASCII, SBCS	CYRILLIC PC-DISPLAY	D6
25424*	ASCII, SBCS	UKRAINE PC-DISPLAY	D8
25425*	ASCII, SBCS	BELARUS PC-DISPLAY	EA
25426	ASCII, SBCS	LATIN-1 PC-DISPLAY	ED
25427	ASCII, SBCS	GREECE PC-DISPLAY	EI

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
25428	ASCII, SBCS	LATIN-2 PC-DISPLAY	EO
25429	ASCII, SBCS	TURKEY PC-DISPLAY	EU
25431	ASCII, SBCS	CYRILLIC PC-DISPLAY	E0
25432	ASCII, SBCS	HEBREW PC-DISPLAY	E8
25433	ASCII, SBCS	TURKEY PC-DISPLAY	FF
25434*	ASCII, SBCS	LATIN-1E PC-DISPLAY	FJ
25435*	ASCII, SBCS	LATIN-9 PC-DISPLAY	FL
25436	ASCII, SBCS	PORTUGAL PC-DISPLAY	FN
25437	ASCII, SBCS	ICELAND PC-DISPLAY	FQ
25438	ASCII, SBCS	HEBREW PC-DISPLAY	FT
25439	ASCII, SBCS	CANADA PC-DISPLAY	FW
25440	ASCII, SBCS	ARABIC PC-DISPLAY	F3
25441	ASCII, SBCS	DEN/NOR PC-DISPLAY	GB
25442	ASCII, SBCS	CYRILLIC PC-DISPLAY	GE
25443*	ASCII, SBCS	HEBREW PC-DISPLAY	GG
25444	ASCII, SBCS	URDU PC-DISPLAY	GK
25445	ASCII, SBCS	GREECE PC-DISPLAY	GS
25448*	ASCII, SBCS	CYRILLIC PC-DISPLAY	G2
25450	ASCII, SBCS	THAILAND PC-DISPLAY	G6
25467	ASCII, SBCS	KOREA PC-DISPLAY	HE
25473	ASCII, SBCS	JAPAN PC-DISPLAY	HO
25477*	ASCII, SBCS	BALTIC ISO-8	HT
25478*	ASCII, SBCS	ESTONIA ISO-8	HV
25479	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) PC-DISPLAY	HX
25480	ASCII, SBCS	TRADITIONAL CHINESE (T-CH) PC-DISPLAY	HZ

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
25488*	ASCII, SBCS	ISO 8859-2	H2
25491*	ASCII, SBCS	ISO 8859-5	H5
25497*	ASCII, SBCS	BALTIC ISO-8	IC
25498*	ASCII, SBCS	ESTONIA ISO-8	IE
25502	ASCII, DBCS	KOREA DB PC-DISPLAY	II
25503	ASCII, DBCS	TRADITIONAL CHINESE (T-CH) PC-DISPLAY	IL
25504	ASCII, DBCS	SIMPLIFIED CHINESE (S-CH) PC-DISPLAY	IN
25505*	ASCII, DBCS	THAILAND PC-DISPLAY	IP
25508	ASCII, MBCS	JAPAN PC-DISPLAY	I2
25510	ASCII, MBCS	KOREA PC-DISPLAY	JB
25512	ASCII, MBCS	SIMPLIFIED CHINESE (S- CH) PC-DISPLAY	JH
25514	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) PC-DISPLAY	JL
25518	ASCII, MBCS	JAPAN PC-DISPLAY	JW
25520	ASCII, MBCS	KOREA PC-DISPLAY	J4
25522	ASCII, MBCS	SIMPLIFIED CHINESE (S- CH) PC-DISPLAY	J7
25524	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) PC-DISPLAY	KG
25525	ASCII, MBCS	KOREA KS PC-DISPLAY	KK
25527	ASCII, DBCS	KOREA KS PC-DISPLAY	KU
25546	ASCII, MBCS	KOREAN TCP	LS
25580	ASCII, SBCS	LATIN-1	LY
25616	ASCII, SBCS	KOREA PC-DISPLAY	ML
25617	ASCII, SBCS	JAPAN PC-DISPLAY	MP
25618	ASCII, SBCS	SIMPLIFIED CHINESE (S- CH) PC-DISPLAY	MS
25619	ASCII, SBCS	TRADITIONAL CHINESE (T-CH) PC-DISPLAY	MV
25664	ASCII, SBCS	KOREA KS PC-DISPLAY	M4

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
25690	ASCII, SBCS	TRADITIONAL CHINESE (T-CH)PC-DISPLAY	NK
25691	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) GB	NO
25702*	ASCII, SBCS	KOREAN MS-WIN	NV
25703*	ASCII, SBCS	ARABIC/FRENCH PC-DISPLAY	NX
28709	EBCDIC, SBCS	TRADITIONAL CHINESE (T-CH)	AH
29109	ASCII, SBCS	USA PC-DISPLAY	CL
29172	EBCDIC, SBCS	BRAZIL	CY
29522	ASCII, SBCS	LATIN-1 PC-DISPLAY	EE
29523	ASCII, SBCS	GREECE PC-DISPLAY	EJ
29524	ASCII, SBCS	LATIN-2 PC-DISPLAY	EP
29525	ASCII, SBCS	TURKEY PC-DISPLAY	EV
29527	ASCII, SBCS	CYRILLIC PC-DISPLAY	E1
29528	ASCII, SBCS	HEBREW PC-DISPLAY	E9
29529	ASCII, SBCS	TURKEY PC-DISPLAY	FG
29532	ASCII, SBCS	PORTUGAL PC-DISPLAY	FO
29533	ASCII, SBCS	ICELAND PC-DISPLAY	FR
29534	ASCII, SBCS	HEBREW PC-DISPLAY	FU
29535	ASCII, SBCS	CANADA PC-DISPLAY	FX
29536	ASCII, SBCS	ARABIC PC-DISPLAY	F4
29537	ASCII, SBCS	DEN/NOR PC-DISPLAY	GC
29540	ASCII, SBCS	URDU PC- DISPLAY	GL
29541	ASCII, SBCS	GREECE PC-DISPLAY	GT
29546	ASCII, SBCS	THAILAND PC-DISPLAY	G7
29614	ASCII, MBCS	JAPAN PC-DISPLAY	JX
29616	ASCII, MBCS	KOREA PC-DISPLAY	J5

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
29618	ASCII, MBCS	SIMPLIFIED CHINESE (S-CH) PC-DISPLAY	J8
29620	ASCII, MBCS	TRADITIONAL CHINESE (T-CH) PC-DISPLAY	KH
29621	ASCII, MBCS	KOREA KS PC	KL
29623	ASCII, DBCS	KOREA KS PC-DISPLAY	KV
29712	ASCII, SBCS	KOREA PC-DISPLAY	MM
29713	ASCII, SBCS	JAPAN PC-DISPLAY	MQ
29714	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) PC-DISPLAY	MT
29715	ASCII, SBCS	TRADITIONAL CHINESE (T-CH) PC-DISPLAY	MW
29760	ASCII, SBCS	KOREA KS PC-DISPLAY	M5
29786*	ASCII, SBCS	SIMPLIFIED CHINESE (S-CH) PC-DISPLAY	NL
32805	EBCDIC, SBCS	JAPAN LATIN	AI
33058	EBCDIC, SBCS	JAPAN	BK
33205	ASCII, SBCS	SWISS PC-DISPLAY	CM
33268	EBCDIC, SBCS	UNITED KINGDOM / PORTUGAL	CZ
33618	ASCII, SBCS	LATIN-1 PC-DISPLAY	EF
33619	ASCII, SBCS	GREECE PC-DISPLAY	EK
33620	ASCII, SBCS	ROECE PC-DISPLAY	EQ
33621	ASCII, SBCS	TURKEY PC-DISPLAY	EW
33623	ASCII, SBCS	CYRILLIC PC-DISPLAY	E2
33624	ASCII, SBCS	HEBREW PC-DISPLAY	FA
33625*	ASCII, SBCS	TURKEY PC-DISPLAY	FH
33632	ASCII, SBCS	ARABIC PC-DISPLAY	F5
33636	ASCII, SBCS	URDU PC-DISPLAY	GM
33637	ASCII, SBCS	GREECE PC-DISPLAY	GU
33665	ASCII, SBCS	JAPAN PC-DISPLAY	HP

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
33698	EBCDIC, MBCS	JAPAN KATAKANA/KANJI	IV
33699	EBCDIC, MBCS	JAPAN LATIN/KANJI	IY
33700	ASCII, MBCS	JAPAN PC- DISPLAY	I3
33717	ASCII, MBCS	KOREA KS PC-DISPLAY	KM
33722	ASCII, MBCS	IBM EUC JAPANESE	K5
37301	ASCII, SBCS	AUSTRIA and GERMANY PC-DISPLAY	CN
37364*	EBCDIC, SBCS	BELGIUM	C0
37716*	ASCII, SBCS	LATIN-2 PC-DISPLAY	ER
37719	ASCII, SBCS	CYRILLIC PC-DISPLAY	E3
37720*	ASCII, SBCS	HEBREW PC-DISPLAY	FB
37728	ASCII, SBCS	ARABIC PC-DISPLAY	F6
37732	ASCII, SBCS	URDU PC-DISPLAY	GN
37733*	ASCII, SBCS	GREECE PC-DISPLAY	GV
37761	ASCII, SBCS	JAPAN PC-DISPLAY	HQ
37796*	ASCII, MBCS	JAPAN PC- DISPLAY	I4
37813	ASCII, MBCS	KOREA KS PC-DISPLAY	KN
41397	ASCII, SBCS	FRANCE PC-DISPLAY	CO
41460	EBCDIC, SBCS	SWISS	C1
41824	ASCII, SBCS	ARABIC PC-DISPLAY	F7
41828	ASCII, SBCS	URDU PC-DISPLAY	GO
45493	ASCII, SBCS	ITALY PC-DISPLAY	CP
45556	EBCDIC, SBCS	SWISS	C2
45920	ASCII, SBCS	ARABIC PC-DISPLAY	F8
49589	ASCII, SBCS	UNITED KINGDOM PC-DISPLAY	CQ
49652	EBCDIC, SBCS	BELGIUM	C3
50016*	ASCII, SBCS	ARABIC PC-DISPLAY	F9
53668	EBCDIC, SBCS	ARABIC EBCDIC - Special	T8
53685	ASCII, SBCS	USA (with MS controls)	OR
53748	EBCDIC, SBCS	INTERNATIONAL	C4

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
54189	ASCII, DBCS	Special - JAPAN DB PC-Data	UB
54191	ASCII, MBCS	Special-JAPAN OPEN	T9
54289	ASCII, SBCS	Special - JAPAN SB PC-Data	UA
57345*	ASCII, MBCS	JAPANESE TCP	RE
61696	EBCDIC, SBCS	GLOBAL	AM
61697	ASCII, SBCS	GLOBAL	AN
61698	ASCII, SBCS	GLOBAL PC-DISPLAY	AO
61699	ASCII, SBCS	GLBL ISO-8	AQ
61700	ASCII, SBCS	GLBL ISO-7	AR
61710	ASCII, SBCS	GLOBAL USE	AS
61711	EBCDIC, SBCS	GLOBAL USE	AT
61712	EBCDIC, SBCS	GLOBAL USE	AU
61952*	UCS-2, DBCS	UNICODE 1.1	RF
61953	UCS-2, DBCS	UNICODE 1.0	RG
61956	UTF-16, DBCS	With mapping of PUA characters as prescribed by Microsoft®	T0
62208*	ASCII, SBCS	HEBREW	RI
62209*	ASCII, SBCS	HEBREW	RJ
62210*	ASCII, SBCS	ISO 8859-8	RK
62211*	EBCDIC, SBCS	HEBREW	RL
62212*	ASCII, SBCS	HEBREW	RM
62213*	ASCII, SBCS	HEBREW	RN
62214*	ASCII, SBCS	HEBREW	RO
62215*	ASCII, SBCS	MS-WIN HEBREW	RP
62216*	ASCII, SBCS	HEBREW	RQ
62217*	ASCII, SBCS	HEBREW	RR
62218*	ASCII, SBCS	ARABIC	RS
62219*	ASCII, SBCS	HEBREW	RT
62220*	ASCII, SBCS	HEBREW	RU
62221*	ASCII, SBCS	HEBREW	RV
62222*	ASCII, SBCS	ISO 8859-8	RW
62223*	ASCII, SBCS	MS-WIN HEBREW	RX
62224*	EBCDIC, SBCS	ARABIC	RY
62225*	ASCII, SBCS	ARABIC	RZ
62226*	ASCII, SBCS	ARABIC - PC	R0
62227*	ASCII, SBCS	ARABIC ISO 8859-6	R1

Description of CCSIDs

CCSID	ENCODING SCHEME	DESCRIPTION	SUFFIX
62228*	ASCII, SBCS	MS-WIN ARABIC	R2
62229*	EBCDIC, SBCS	HEBREW	R3
62230*	ASCII, SBCS	HEBREW	R4
62231*	ASCII, SBCS	HEBREW	R5
62232*	ASCII, SBCS	ISO 8859-8	R6
62233*	EBCDIC, SBCS	ARABIC	R7
62234*	EBCDIC, SBCS	ARABIC	R8
62235*	EBCDIC, SBCS	HEBREW	R9
62236*	ASCII, SBCS	HEBREW	SA
62237*	ASCII, SBCS	MS-WIN HEBREW	SB
62238*	ASCII, SBCS	ISO 8859-8	SC
62239*	ASCII, SBCS	MS-WIN HEBREW	SD
62240*	EBCDIC, SBCS	HEBREW	SE
62241*	ASCII, SBCS	HEBREW	SF
62242*	ASCII, SBCS	HEBREW	SG
62243*	ASCII, SBCS	ISO 8859-8	SH
62244*	ASCII, SBCS	MS-WIN HEBREW	SI
62245*	EBCDIC, SBCS	HEBREW	SJ
62246*	ASCII, SBCS	ARABIC - PC	SK
62247*	ASCII, SBCS	ARABIC - PC	SL
62248*	ASCII, SBCS	ARABIC - PC	SM
62249*	ASCII, SBCS	ARABIC - PC	SN
62250*	EBCDIC, SBCS	ARABIC	SO
62337	ASCII, SBCS	Special - JAPAN SB PC-Data	UD
62381	ASCII, DBCS	Special - JAPAN DB PC-Data	UE
62383	ASCII, MBCS	Special-JAPAN OPEN	UC

Note: CCSIDs that have an asterisk (*) in column one are not supported by z/OS support for Unicode.

EBCDIC Conversion Table

The following table lists the techniques supported between EBCDIC CCSIDs.

Table 46. EBCDIC Conversion Table

FROM-CCSID	TO-CCSID	Technique Supported
00037	00256	R,E
	00273	R
	00275	R
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E
	00425	R,E
00037	00437	R,E
	00500	R,E,L
	00720	R
	00737	R
	00775	R
	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00037	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00866	R
	00869	R
	00870	R,E
	00871	R,E
	00874	R,E
	00875	R,E
	00880	R,E
	00897	R,E
	00901	R,E
00037	00902	R,E
	00903	R
	00904	E
	00905	R,E
	00912	R
	00914	R
	00915	R
	00916	R
	00920	R
	00921	R
	00922	R
	00923	R,E
	00924	R,E,L
	01009	E
00037	01025	R,E
	01026	R,E
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01089	R,E
	01097	R,E
	01100	R
	01112	R,E
	01114	E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00037	01115	E
	01122	R
	01123	R,E
	01124	R,E
	01126	E
	01130	R
	01131	R,E
	01132	R
	01137	E
	01140	E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
00037	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01250	R
	01251	R
	01252	R,E
	01253	R
	01254	R
	01255	R
	01257	R
	01258	R,E
	01275	R
	01280	R
	01281	R
	01283	R
	04909	R,E
	05210	E
	05348	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00256	00037	R,E
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	E
	00297	R
	00367	E
	00420	R
	00423	R
	00424	R
	00437	R,E
	00500	R,E
00256	00737	R
	00775	R,E
	00819	R
	00833	E
	00836	E
	00838	E
	00850	R,E
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00256	00866	E,C
	00869	R
	00870	R,E
	00871	R
	00875	R
	00880	R
	00905	R
	01025	R
	01026	R
	01027	E
	01112	R
	01122	R
	01200 (13488)	R
	01208	R
	01251	R,E
	01252	R,E
	01275	R
00259	00437	E
	00808	E
	00850	E
	00851	E
	00852	E
	00855	R,E
	00856	E
	00857	E
	00858	E
	00860	E
	00861	E
	00862	E
	00863	E
	00864	E
	00865	E
	00866	E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00259	00867	E
	00869	E
	00872	E
	00874	E
	00899	E
	00901	E
	00902	E
	00915	R,E
	01051	E
	01098	R,E
	01161	E
	01162	E
	01200 (13488)	R,E
	01208	R,E
	01250	E
	01251	E
00259	01252	E
	01253	E
	01254	E
	01255	E
	01256	E
	01257	E
	01258	E
	05348	E
00273	00037	R,E
	00256	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R
	00775	R

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00273	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00856	E
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
00273	00864	R,E
	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E,L
00273	01009	E
	01025	R
	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01100	R
	01112	R
	01122	R
	01140	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00273	01141	E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01250	R,E
	01252	R,E
	01275	R
	05348	R,E
00274	00500	R,L
	00819	R,E,L
	00850	R,E
	01047	R,L
	01148	R,E,L
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
00275	00037	R
	00437	R,E
	00500	R,L
	00819	R,E,L
	00850	R,E
	01047	R,L
	01148	R,E,L
	01252	R,E
	05348	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00277	00037	R,E
	00256	R
	00273	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R
	00775	R,E
00277	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
00277	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E
	01009	E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00277	01025	R
	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01100	R
	01112	R
	01122	R
	01140	R,E
	01141	R,E
00277	01142	E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01275	R
	05348	R,E
00278	00037	R,E
	00256	R
	00273	R
	00277	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R
	00775	R

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00278	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
00278	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E,L
	01009	E
00278	01025	R
	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01100	R
	01112	R
	01122	R
	01140	R,E
	01141	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00278	01142	R,E
	01143	E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01275	R
	05348	R,E
00280	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R
	00775	R,E
00280	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00280	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E,L
	01009	E
00280	01025	R
	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01100	R
	01112	R
	01122	R
	01140	R,E
	01141	R,E
00280	01142	R,E
	01143	R,E
	01144	E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01275	R
	05348	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00281	00500	R,E,L
	00819	R,E,L
	01047	R,L
	01148	R,E,L
00282	00500	R,L
	00819	R,E,L
	01047	R,L
	01051	E
	01148	R,E,L
00284	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R
	00775	R
00284	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00284	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E,L
	01009	E
	01025	R
00284	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01100	R
	01112	R
	01122	R
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
00284	01144	R,E
	01145	E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01275	R
	05348	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00285	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R
	00775	R,E
00285	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
00285	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E,L
	01025	R

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00285	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01100	R
	01112	R
	01122	R
	01140	R,E
	01141	R,E
	01142	R,E
00285	01143	R,E
	01144	R,E
	01145	R,E
	01146	E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01275	R
	05348	R,E
00290	00037	R,E
	00256	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00737	E
	00775	E
	00819	E,L

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00290	00833	R,E
	00836	R,E
	00850	R,E
	00852	R,E
	00855	R,E
	00857	R,E
	00858	E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00870	R,E
	00871	R,E
00290	00895	E
	00896	E
	00897	E
	01009	E
	01025	R,E
	01026	R,E
	01027	R,L
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,E,L
	01088	R
	01112	R
	01122	R
	01148	R,E,L
	01200 (13488)	R,L
	01208	R,L
	01252	E
	05348	E
00293	01200 (13488)	R,E
	01208	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00297	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R
	00775	R,E
00297	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
00297	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E,L
	01009	E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00297	01025	R
	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01100	R
	01112	R
	01122	R
	01140	R,E
	01141	R,E
00297	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01275	R
	05348	R,E
00300	00301	E
	00941	E
	01200 (13488)	R
	01208	R
	01351	E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00420	00037	R,E
	00256	R
	00367	E
	00424	R
	00425	C
	00437	R,E
	00500	R,E
	00720	C
	00737	R
	00775	R
	00819	R
	00850	R
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
00420	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	01008	R
	01046	C
	01051	E
	01089	C
	01098	R
	01112	R
	01122	R
	01127	R
	01200 (13488)	R,C,L
	01208	R,C,L
	01252	R
	01256	C
00420	05352	C
	09238	E
	17248	R,E
00421	00367	E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00423	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00437	R,E
	00500	R,E
	00737	R,E
	00775	R,E
	00813	R
00423	00819	R
	00838	R
	00850	R
	00851	R
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00869	R
	00870	R
	00871	R
00423	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	01009	E
	01025	R
	01026	R
	01027	R
	01041	R

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00423	01042	R
	01043	R
	01051	E
	01112	R
	01122	R
	01200 (13488)	R
	01208	R
	01252	R
	01253	R,E
	01280	R
	09061	R,E
00424	00037	R,E
	00256	R
	00367	E
	00420	R
	00437	R,E
	00500	R,E
	00737	R
	00775	R
	00803	R
	00819	R
	00836	E
	00850	R,E
	00852	R,E
	00856	R
	00857	R,E
	00860	R,E
00424	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00867	R
	00916	R,E
	01051	E
	01112	R
	01122	R
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01255	R,E
	05351	R,E
	09048	R

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00425	00037	R,E
	00420	C
	00500	R,E
	00720	E
	00819	R,E
	00864	C
	01046	C
	01089	E
	01140	R,E
	01148	R,E
	01200 (17584)	R,C,L
	01208	R,C,L
	01256	E
	05348	R,E
	05352	R,E
	16804	C
00500	00037	R,E,L
	00256	R,E
	00273	R,E,L
	00274	R,L
	00275	R,L
	00277	R,E,L
	00278	R,E,L
	00280	R,E,L
	00281	R,E,L
	00282	R,L
	00284	R,E,L
	00285	R,E,L
	00290	R,E,L
	00297	R,E,L
	00367	E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00500	00420	R,E
	00423	R,E
	00424	R,E
	00425	R,E
	00437	R,E
	00737	R,E
	00775	R,E
	00813	R,E
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00851	R
	00852	R,E
00500	00855	R
	00856	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00866	E
	00869	R,E
	00870	R,E
	00871	R,E,L
	00875	R,E
00500	00880	R,E
	00891	E
	00895	E
	00897	E
	00901	R,E
	00902	R,E
	00903	E
	00904	E
	00905	R,E
	00912	R,E
	00914	R
	00915	R
	00916	R,E
	00920	R,E
	00921	R

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00500	00922	R
	00923	R,E
	00924	R,E,L
	01004	R
	01009	E
	01010	E
	01011	E
	01012	E
	01013	E
	01014	E
	01015	E
	01016	E
	01017	E
	01018	E
	01019	E
00500	01020	E
	01021	E
	01023	E
	01025	R,E
	01026	R,E
	01027	R,E,L
	01040	R,E
	01041	R,E
	01042	R,E
	01043	R,E
	01046	E
	01047	R,L
	01051	R,E
	01088	R,E
	01089	R,E
00500	01097	R,E
	01100	R,E
	01101	E
	01102	E
	01103	E
	01104	E
	01105	E
	01106	E
	01107	E
	01112	R,E
	01114	E
	01115	E
	01122	R
	01123	R,E
	01124	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00500	01125	R,E
	01126	E
	01129	R,E
	01130	R,E
	01131	R
	01132	R,E
	01133	R,E
	01137	E
	01140	R,E,L
	01141	R,E,L
	01142	R,E,L
	01143	R,E,L
	01144	R,E,L
	01145	R,E,L
	01146	R,E,L
00500	01147	R,E,L
	01148	E
	01149	R,E,L
	01200 (13488)	R,L
	01208	R,L
	01250	R,E
	01251	R,E
	01252	R,E
	01253	R,E
	01254	R,E
	01255	R,E
	01256	R,E
00500	01257	R
	01258	R,E
	01275	R
	01280	R
	01281	R
	01282	R
	01283	R
	04909	R,E
	05348	R,E
	05350	R
	09049	E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00803	00367	E
	00424	R
	00819	R,E
	00850	R,E
	00856	R
	00862	R,E
	00916	R,E
	01200 (13488)	R
	01208	R
	01252	R,E
	01255	R,E
00833	00037	R,E
	00256	E
	00273	R,E
	00277	R,E
	00278	R,E
	00278	R
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E
	00737	E
00833	00775	E
	00819	E
	00836	R,E
	00850	R,E
	00852	R,E
	00855	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00870	R,E
	00871	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00833	00891	E
	01009	E
	01025	R,E
	01026	R,E
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,E,L
	01088	R,E
	01112	R
	01122	R
	01126	E
	01200 (13488)	R,L
	01208	R,L
	01252	E
00834	00926	E
	00951	E
	00971	E
	01200 (13488)	R,E
	01208	R,E
	01362	E
	04930	E
00835	00927	E
	00947	E
	01200 (13488)	E
	01208	E
00836	00037	R,E
	00256	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00424	E
	00437	E
	00500	R,E
	00737	E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00836	00775	E
	00819	E
	00833	R,E
	00850	R,E
	00852	R
	00855	R
	00857	R
	00870	R
	00871	R,E
	00875	R,E
	00903	E
	01009	E
	01025	R
	01026	R
	01027	R,E
00836	01040	R
	01041	R
	01042	R,E
	01043	R
	01047	R,E,L
	01088	R
	01112	R
	01114	E
	01115	E
	01122	R
	01200 (13488)	R,L
	01208	R,L
	01252	E
00837	00928	E
	01200 (13488)	R,E
	01208	R,E
	01380	E
	01382	E
	01385	E
	04933	E
	13125	E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00838	00037	E
	00256	E
	00273	E
	00277	E
	00278	E
	00280	E
	00284	E
	00285	E
	00297	E
	00367	E
	00423	R
	00437	R,E
	00500	E
	00737	E
	00775	E
00838	00813	R
	00819	R
	00850	R,E
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00869	R
	00870	R
	00871	E
	00874	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00838	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
	01043	R
	01051	E
	01112	R
	01122	R
	01161	R,E
	01200 (13488)	E,L
	01208	E,L
	01252	E
00870	00037	R,E
	00256	R,E
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E
	00737	R

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00870	00775	R,E
	00813	R
	00819	R
	00833	R,E
	00836	R
	00838	R
	00850	R,E
	00852	R,E
	00855	R
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
00870	00865	R,E
	00866	R
	00869	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00915	R
	00916	R
	00920	R
	01009	E
	01025	R
	01026	R

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00870	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R
	01051	E
	01088	R
	01112	R
	01122	R
	01147	R,E
	01200 (13488)	R,L
	01208	R,L
	01250	R,E
	01252	R
	01282	R
00870	05346	R,E
	09044	R,E
00871	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	R

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00871	00775	R
	00813	R
	00819	R,L
	00833	R,E
	00836	R,E
	00838	E
	00850	R,E,C,L
	00852	R,E
	00855	R
	00857	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
00871	00864	R,E
	00865	R,E
	00869	R
	00870	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E,L
	01009	E
00871	01025	R
	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01051	R,E
	01088	R
	01112	R
	01122	R
	01140	R,E
	01141	R,E
	01142	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00871	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	E
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01275	R
	05348	R,E
00875	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E
	00737	R,E
	00775	R,E
00875	00813	R,L
	00819	R
	00836	R,E
	00838	R
	00850	R
	00851	R
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00869	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00875	00870	R
	00871	R
	00874	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00920	R
	01009	E
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
00875	01043	R
	01047	R,L
	01051	E
	01088	R
	01112	R
	01122	R
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01253	R,E
	01280	R
	01287	R,E
	04909	R,E
	05349	R,E
	09061	R,E
00880	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E
	00737	R
	00775	R

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00880	00813	R
	00819	R
	00838	R
	00850	R,E
	00852	R,E
	00855	R
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	00866	E
	00869	R
00880	00870	R
	00871	R
	00874	R
	00875	R
	00878	R,E
	00897	R
	00903	R
	00912	R
	00915	R
	00916	R
	00920	R
	01009	E
	01025	R,E
	01026	R
00880	01027	R
	01041	R
	01042	R
	01043	R
	01051	E
	01112	R
	01122	R
	01200 (13488)	R,L
	01208	R,L
	01251	R,E
	01252	R
	01283	R
	05347	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00905	00037	R,E
	00256	R
	00367	E
	00437	R,E
	00500	R,E
	00737	R
	00775	R,E
	00819	R
	00850	R,E
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
00905	00863	R,E
	00864	R,E
	00865	R,E
	00920	R
	01026	R
	01051	E
	01112	R
	01122	R
	01200 (13488)	R
	01208	R
	01252	R
	01254	R,E
	01281	R
00918	00367	E
	00864	R
	00868	R
	01006	R
	01200 (13488)	R,E
	01208	R,E

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00924	00037	R,E,L
	00273	R,E,L
	00277	R,E
	00278	R,E,L
	00280	R,E,L
	00284	R,E,L
	00285	R,E,L
	00297	R,E,L
	00367	E
	00437	R,E
	00500	R,E,L
	00808	R,E
	00819	R,E
	00848	R,E
	00849	R,E
00924	00850	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00865	R,E
	00871	R,E,L
	00872	R,E
	00901	R,E
	00902	R,E
	00923	R,L
	01047	R,E,L
	01051	R,E
	01140	R,E,L
	01141	R,E,L
	01142	R,E,L
00924	01143	R,E,L
	01144	R,E,L
	01145	R,E,L
	01146	R,E,L
	01147	R,E,L
	01148	R,E,L
	01149	R,E,L
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E

EBCDIC Conversion Table

Table 46. EBCDIC Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00924	01163	R,E
	01164	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E,L
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	09238	R,E
	16804	R,E
	17248	R,E
01025	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E
	00737	R

EBCDIC Conversion Table

01025	00775	R
	00808	R,E
	00813	R
	00819	R
	00833	R,E
	00836	R
	00838	R
	00849	R,E
	00850	R,E
	00852	R,E
	00855	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
01025	00863	R,E
	00864	R,E
	00865	R,E
	00866	R,E
	00869	R
	00870	R
	00871	R
	00872	R,E
	00874	R
	00875	R
	00878	R,E
	00880	R,E
	00897	R
	00903	R
	00912	R
01025	00915	R,E
	00916	R
	00920	R
	01009	E
	01026	R
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01051	R
	01088	R
	01112	R
	01122	R
	01131	R

EBCDIC Conversion Table

01025	01167	R,E
	01200 (13488)	R,L
	01208	R,L
	01251	R,E
	01252	R
	01283	R
	05347	R,E
01026	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	R,E
	00737	R
01026	00775	R,E
	00813	R
	00819	R
	00833	R,E
	00836	R
	00838	R
	00850	R,E
	00852	R,E
	00855	R
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E

EBCDIC Conversion Table

01026	00865	R,E
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00905	R
	00912	R
	00916	R
	00920	R,L
	01009	E
	01025	R
	01027	R,E
01026	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01088	R
	01112	R
	01122	R
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01254	R,E,L
	01281	R
	01288	R,E
	05350	R,E
	09049	R
01027	00037	R,E
	00256	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,L
	00297	R,E
	00367	E
	00423	R
	00437	R,E
	00500	R,E,L
	00737	E

EBCDIC Conversion Table

01027	00775	E
	00813	R
	00819	R,E,L
	00833	R,E
	00836	R,E
	00838	R
	00850	R,E
	00852	R,E
	00855	R,E
	00857	R,E
	00858	E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
01027	00864	R,E
	00865	R,E
	00869	R
	00870	R,E
	00871	R,E
	00874	R
	00875	R
	00880	R
	00895	E
	00896	E
	00897	E
	00903	R
	00912	R
	00916	R
	01025	R,E
01027	01026	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,L
	01088	R
	01112	R
	01122	R
	01148	R,E,L
	01200 (13488)	R,L
	01208	R,L
	01252	E
	05348	E

EBCDIC Conversion Table

01047	00037	R,L
	00273	R,L
	00274	R,L
	00275	R,L
	00277	R,L
	00278	R,L
	00280	R,L
	00281	R,L
	00282	R,L
	00284	R,L
	00285	R,L
	00290	R,E,L
	00297	R,L
	00367	E
	00437	R,E
01047	00500	R,L
	00819	R,L
	00833	R,E,L
	00836	R,E,L
	00850	R,C,L
	00852	R
	00858	R,E,L
	00870	R
	00871	R,L
	00875	R,L
	00912	R
	00923	R,E
	00924	R,E,L
	01026	R,L
	01027	R,L
01047	01140	R,E,L
	01141	R,E,L
	01142	R,E,L
	01143	R,E,L
	01144	R,E,L
	01145	R,E,L
	01146	R,E,L
	01147	R,E,L
	01148	R,E,L
	01149	R,E,L
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01254	R,E
	05348	R

EBCDIC Conversion Table

01097	00037	R,E
	00367	E
	00437	R,E
	00500	R,E
	00737	R
	00775	R,E
	00819	R
	00850	R
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
01097	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	01051	R,E
	01098	R,E
	01112	R
	01122	R
	01200 (13488)	R,E
	01208	R,E
	01252	R
01112	00037	R,E
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R
	00297	R
	00367	E
	00420	R
	00423	R
	00424	R
	00500	R,E

EBCDIC Conversion Table

01112	00775	R
	00819	R,E
	00833	R
	00836	R
	00838	R
	00850	R
	00870	R
	00871	R
	00875	R
	00880	R
	00905	R
	00914	R,E
	00921	R,E
	00922	R
	01025	R
01112	01026	R
	01027	R
	01097	R
	01122	R
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01257	R,E
	05353	R,E
01122	00037	R
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R
	00297	R
	00367	E
	00420	R
	00423	R
	00424	R
	00500	R

EBCDIC Conversion Table

01122	00775	R
	00819	R,E
	00833	R
	00836	R
	00838	R
	00850	R
	00870	R
	00871	R
	00875	R
	00880	R
	00905	R
	00914	R,E
	00921	R
	00922	R,E
	01025	R
01122	01026	R
	01027	R
	01097	R
	01112	R
	01200 (13488)	R,L
	01208	R,L
	01252	R,E
	01257	R,E
	05353	R,E
01123	00037	R,E
	00367	E
	00500	R,E
	00819	R,E
	00848	R,E
	01124	R,E
	01125	R,E
	01148	R,E
	01168	R,E
	01200 (13488)	R,L
	01208	R,L
	01251	R,E
	01252	R,E
	01283	R
	05347	R,E

EBCDIC Conversion Table

01130	00037	R
	00500	R,E
	00819	R,E
	00850	R,E
	01129	R,E
	01163	R,E
	01200 (13488)	R
	01208	R
	01252	R,E
	01258	R,E
	05354	R,E
01132	00037	R
	00500	R,E
	00819	R,E
	00850	R,E
	01133	R,E
	01200 (13488)	R
	01200 (17584)	R
	01208	R
	01252	R,E
01137	00037	E
	00500	E
	00806	E
	00819	E
	01200 (13488)	R
	01208	R
	01252	R,E
01140	00037	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00425	R,E
	00437	R,E
	00500	R,E,L
	00808	R,E
	00819	R,E,L
	00850	R,E,L

EBCDIC Conversion Table

01140	00858	R,E,L
	00860	R,E
	00863	R,E
	00871	R,E
	00872	R,E
	00901	R,E
	00902	R,E
	00923	R,E
	00924	R,E,L
	01013	E
	01047	R,E,L
	01051	R,E
	01141	R
	01142	R
	01143	R
01140	01144	R
	01145	R
	01146	R
	01147	R
	01148	R,L
	01149	R
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
01140	01164	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	16804	R,E
	17248	R,E

EBCDIC Conversion Table

01141	00037	R,E
	00273	E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00863	R,E
01141	00871	R,E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01142	R
	01143	R
	01144	R
	01145	R
	01146	R
	01147	R
	01148	R,L
	01149	R

EBCDIC Conversion Table

01141	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E
01142	00037	R,E
	00273	R,E
	00277	E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00863	R,E

EBCDIC Conversion Table

01142	00865	R,E
	00871	R,E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01141	R
	01143	R
	01144	R
	01145	R
	01146	R
	01147	R
	01148	R,L
01142	01149	R
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E

EBCDIC Conversion Table

01143	00037	R,E
	00273	R,E
	00277	R,E
	00278	E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00863	R,E
01143	00865	R,E
	00871	R,E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01141	R
	01142	R
	01144	R
	01145	R
	01146	R
	01147	R
	01148	R,L

EBCDIC Conversion Table

01143	01149	R
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E
01144	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00863	R,E

EBCDIC Conversion Table

01144	00871	R,E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01141	R
	01142	R
	01143	R
	01145	R
	01146	R
	01147	R
	01148	R,L
	01149	R
01144	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E

EBCDIC Conversion Table

01145	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00860	R,E
01145	00863	R,E
	00871	R,E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01141	R
	01142	R
	01143	R
	01144	R
	01146	R
	01147	R
	01148	R,L

EBCDIC Conversion Table

01145	01149	R
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E
01146	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00860	R,E

EBCDIC Conversion Table

01146	00863	R,E
	00871	R,E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01141	R
	01142	R
	01143	R
	01144	R
	01145	R
	01147	R
	01148	R,L
01146	01149	R
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E

EBCDIC Conversion Table

01147	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00863	R,E
01147	00870	R,E
	00871	R,E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01141	R
	01142	R
	01143	R
	01144	R
	01145	R
	01146	R
	01148	R,L

EBCDIC Conversion Table

01147	01149	R
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E
01148	00037	R,E
	00273	R,E
	00274	R,E,L
	00275	R,E,L
	00277	R,E
	00278	R,E
	00280	R,E
	00281	R,E,L
	00282	R,E,L
	00284	R,E
	00285	R,E
	00290	R,E,L
	00297	R,E
	00367	E
	00425	R,E

EBCDIC Conversion Table

01148	00437	R,E
	00500	E
	00808	R,E
	00819	R,E,L
	00848	R,E
	00849	R,E
	00850	R,E,L
	00858	R,E,L
	00860	R,E
	00861	R,E
	00863	R,E
	00865	R,E
	00867	R,E
	00871	R,E
	00872	R,E
01148	00901	R,E
	00902	R,E
	00912	E
	00916	E
	00920	E
	00923	R,E
	00924	R,E,L
	01027	R,E,L
	01047	R,E,L
	01051	R,E
	01089	E
	01123	R,E
	01140	R,L
	01141	R,L
	01142	R,L
01148	01143	R,L
	01144	R,L
	01145	R,L
	01146	R,L
	01147	R,L
	01149	R,L
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01158	R,E
	01159	R,E
	01160	R,E
	01161	R,E

EBCDIC Conversion Table

01148	01162	R,E
	01163	R,E
	01164	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	01281	R,E
	04899	R,E
	04909	R,E
	04971	R,E
	05123	E
	05210	R,E
	05346	R,E
	05347	R,E
	05348	R,E
01148	05349	R,E
	05350	R,E
	05351	R,E
	05352	R,E
	05353	R,E
	05354	R,E
	08482	R,E
	09044	R,E
	09048	R,E
	09049	R,E
	09061	R,E
	09238	R,E
	12712	R,E
	16804	R,E
	17248	R,E
01149	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E,L
	00819	R,E,L
	00850	R,E,L
	00858	R,E,L
	00861	R,E

EBCDIC Conversion Table

01149	00863	R,E
	00871	E
	00872	R,E
	00923	R,E
	00924	R,E,L
	01047	R,E,L
	01051	R,E
	01140	R
	01141	R
	01142	R
	01143	R
	01144	R
	01145	R
	01146	R
	01147	R
01149	01148	R,L
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E

EBCDIC Conversion Table

01153	00808	R,E
	00819	R,E
	00850	R,E
	00852	R,E
	00858	R,E
	00859	R,E
	00867	R,E
	00872	R,E
	00912	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
01153	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01282	R,E
	04909	R,E
01153	04971	R,E
	05346	R,E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E

EBCDIC Conversion Table

01154	00808	R,E
	00819	R,E
	00848	R
	00849	R,E
	00858	R,E
	00859	R,E
	00867	R,E
	00872	R,E
	00915	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
01154	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01153	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01283	R,E
	04909	R,E
01154	04971	R,E
	05123	E
	05347	R,E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	16804	R,E
	17248	R,E

EBCDIC Conversion Table

01155	00819	R,E
	00858	R,E
	00859	R,E
	00867	R,E
	00872	R,E
	00920	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
01155	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01281	R,E
	04909	R,E
	04971	R,E
	05348	R,E
	05350	R,E
01155	09044	R,E
	09049	R,E
	09061	R,E
	16804	R,E
	17248	R,E

EBCDIC Conversion Table

01156	00819	R,E
	00858	R,E
	00859	R,E
	00901	R,E
	00902	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
01156	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01157	R,E
	01160	R,E
	01200 (17584)	R,L
	01208	R,L
	04971	R,E
	05123	E
	05348	R,E
	05353	R,E
	12712	R,E
	16804	R,E
01157	00819	R,E
	00858	R,E
	00859	R,E
	00901	R,E
	00902	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E

EBCDIC Conversion Table

01157	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01160	R,E
	01200 (17584)	R,L
	01208	R,L
	04971	R,E
	05123	E
	05348	R,E
	05353	R,E
	12712	R,E
	16804	R,E
01158	00808	R
	00819	R,E
	00848	R,E
	00849	R
	00923	R,E
	01124	E
	01148	R,E
	01200 (17584)	R,L
	01208	R,L
	05347	R,E
	05348	R,E
01159	01148	R,E
	01200 (17584)	R,L
	01208	R,L
	05210	E
01160	00819	R,E
	00858	R,E
	00859	R,E
	00867	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01153	R,E

EBCDIC Conversion Table

01160	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	04909	R,E
	04971	R,E
	05123	E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E
01164	00819	R,E
	00858	R,E
	00859	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01148	R,E
	01163	R,E
	01200 (17584)	R
	01208	R
	05348	R,E
	05354	R,E
01165	01200 (17584)	R,L
	01208	R,L
01166	01200 (17584)	R
	01208	R
01374	17584	R
	61596	R
01385	17584	R
04133	00367	E
	01200 (13488)	R
	01208	R
04369	00367	E
	01200 (13488)	R
	01208	R
04370	01200 (17584)	R
	01208	R
04371	00367	E
	01200 (13488)	R
	01208	R

EBCDIC Conversion Table

04373	00367	E
	01200 (13488)	R
	01208	R
04374	00367	E
	01200 (13488)	R
	01208	R
04376	00367	E
	01200 (13488)	R
	01208	R
04378	00367	E
	01200 (13488)	R
	01208	R
04380	00367	E
	01200 (13488)	R
	01208	R
04381	00367	E
	01200 (13488)	R
	01208	R
04386	00367	E
	01200 (13488)	R,C
	01208	R,C
04393	01200 (13488)	R
	01208	R
04396	01200 (13488)	R
	01200 (17584)	R
	01208	R
04516	00367	E
	01200 (13488)	R,C
	01208	R,C
04517	01200(21680)	C
04519	00367	E
	01200 (13488)	R
	01208	R
04520	00367	E
	01200 (13488)	R
	01208	R
04596	00367	E
	01200 (13488)	R
	01208	R
04899	00867	R,E
	01148	R,E
	01200 (17584)	R
	01208	R
	05012	R,E
	05351	R,E
	09048	R,E
	12712	R,E

EBCDIC Conversion Table

04929	00367	E
	01200 (13488)	R
	01208	R
04930	00834	E
	00951	E
	01200 (13488)	R
	01200 (17584)	R
	01208	R
	01362	E
04931	01200 (13488)	E
	01208	E
04932	00367	E
	01200 (13488)	R
	01208	R
04933	00837	E
	01200 (13488)	R
	01200 (17584)	R
	01208	R
	01380	E
	01385	E
04934	00367	E
	01200 (13488)	E
	01208	E
04966	00367	E
	01200 (13488)	R
	01208	R
04967	00367	E
	01200 (13488)	R
	01208	R
04971	00858	R,E
	00859	R,E
	00867	R,E
	00923	R,E
	00924	R,E,L
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E

EBCDIC Conversion Table

04971	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	01280	R,E
	04909	R,E,L
	05348	R,E
	05349	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	17248	R,E
04976	00367	E
	01200 (13488)	R
	01208	R
05014	00367	E
	01200 (13488)	R,E
	01208	R,E
05123	00858	E
	01140	E
	01141	E
	01142	E
	01143	E
	01144	E
	01145	E
	01146	E
	01147	E
	01148	E
	01149	E
	01154	E
	01156	E
	01157	E
	01160	E
	01200 (17584)	R,L
	01208	R,L
	05348	E
	08482	R
05143	00367	E
	01200 (13488)	R
	01208	R
05470	17584	R
	61956	R
05472	17584	R

EBCDIC Conversion Table

08229	00367	E
	01200 (13488)	R,C
	01208	R,C
08448	00367	E
	01200 (13488)	R
	01208	R
08482	00858	R,E
	01148	R,E
	01200 (17584)	R,L
	01208	R
	05123	R
	05348	E
08612	01200 (13488)	R
	01208	R
08692	00367	E
	01200 (13488)	R
	01208	R
09025	00367	E
	01200 (13488)	R
	01208	R
09026	01200 (13488)	R,E
	01208	R,E
09027	01200 (17584)	E
	01208	E
	21427	E
09028	00367	E
	01200 (13488)	R
	01208	R
09444	01200(17584)	R,C
	17584	R
	61956	R
12544	00367	E

EBCDIC Conversion Table

12712	00862	R,E
	00867	R,E
	00916	R,E
	01148	R,E
	01156	R,E
	01157	R,E
	01200 (17584)	R,L
	01208	R,L
	04899	R,E
	05012	R,E
	05348	R,E
	05351	R,E
	09044	R,E
	09048	R,E
	09049	R,E
	09447	R,E
	16804	R,E
	17248	R,E
12788	00367	E
	01200 (13488)	R
	01208	R
13121	01126	E
	01200 (17584)	R,L
	01208	R
13124	01200 (13488)	R,L
	01208	R,L
13125	00837	E
	01200 (13488)	R
	01208	R
	01380	E
	01385	E
16421	00367	E
	01200 (13488)	R
	01208	R
16684	01200 (13488)	R
	01200 (17584)	R
	01208	R

EBCDIC Conversion Table

16804	00425	C
	00858	R,E
	00859	R,E
	00867	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01148	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01200 (17584)	R,L
	01208	R,L
	05104	R,E
	05348	R,E
	05352	C
	09044	R,E
	09049	R,E
	09238	C
	12712	R,E
	17248	R,E
16884	00367	E
	01200 (13488)	R
	01208	R
20517	00367	E
	01200 (13488)	R
	01208	R
20780	01200 (17584)	R
	01208	R
20980	00367	E
	01200 (13488)	R
	01208	R
21314	01200 (13488)	R,E
	01208	R,E
21317	01200 (13488)	R,E
	01208	R,E
24613	00367	E
	01200 (13488)	R
	01208	R
25076	00367	E
	01200 (13488)	R
	01208	R

EBCDIC Conversion Table

28709	00437	R,E
	00737	E
	00775	E
	00852	R,E
	00857	R,E
	00860	R,E
	00861	R,E
	00862	R,E
	00863	R,E
	00864	R,E
	00865	R,E
	01114	E
	01200 (13488)	R
	01208	R
29172	00367	E
	01200 (13488)	R
	01208	R
32805	00367	E
33058	00367	E
33268	00367	E
	01200 (13488)	R
	01208	R
41460	00367	E
	01200 (13488)	R
	01208	R
45556	00367	E
	01200 (13488)	R
	01208	R
49652	00367	E
	01200 (13488)	R
	01208	R
53668	01200(13488)	R,L
53748	00367	E
	01200 (13488)	R
	01208	R
61696	00367	E
	01200 (13488)	R
	01208	R
61711	00367	E
	01200 (13488)	R
	01208	R
61712	00367	E
	01200 (13488)	R
	01208	R
61956	01374	E
	05470	E
	09444	E

I

ASCII Conversion Table

The following table lists the techniques supported between ASCII CCSIDs.

Table 47. ASCII Conversion Table

FROM-CCSID	TO-CCSID	Technique Supported
00301	00300	E
	00941	E
	01200 (13488)	R
	01208	R
	01351	E
00367	00037	E
	00256	E
	00273	E
	00277	E
	00278	E
	00280	E
	00284	E
	00285	E
	00290	E
	00297	E
	00420	E
	00421	E
	00423	E
	00424	E
	00437	E
00367	00500	E
	00803	E
	00813	E
	00819	E
	00833	E
	00836	E
	00838	E
	00850	E
	00851	E
	00852	E
	00853	E
	00855	E
	00856	E
	00857	E
	00858	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00367	00860	E
	00861	E
	00862	E
	00863	E
	00864	E
	00865	E
	00866	E
	00868	E
	00869	E
	00870	E
	00871	E
	00874	E
	00875	E
	00880	E
	00891	E
00367	00895	E
	00896	E
	00897	E
	00903	E
	00904	E
	00905	E
	00912	E
	00915	E
	00916	E
	00918	E
	00920	E
	00921	E
	00922	E
	00923	E
	00924	E
00367	01004	E
	01006	E
	01008	E
	01009	R
	01010	E
	01011	E
	01012	E
	01013	E
	01014	E
	01015	E
	01016	E
	01017	E
	01018	E
	01019	E
	01020	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00367	01021	E
	01023	E
	01025	E
	01026	E
	01027	E
	01040	E
	01041	E
	01042	E
	01043	E
	01046	E
	01047	E
	01051	E
	01088	E
	01089	E
	01097	E
00367	01098	E
	01100	E
	01101	E
	01102	E
	01103	E
	01104	E
	01105	E
	01106	E
	01107	E
	01112	E
	01114	E
	01115	E
	01122	E
	01123	E
	01124	E
	01125	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00367	01126	E
	01131	E
	01140	E
	01141	E
	01142	E
	01143	E
	01144	E
	01145	E
	01146	E
	01147	E
	01148	E
	01149	E
	01200 (13488)	R
	01208	R
	01250	E
	01251	E
00367	01252	E
	01253	E
	01254	E
	01255	E
	01256	E
	01257	E
	01275	E
	01276	E
	01277	E
	01280	E
	01281	E
	01282	E
	01283	E
	04133	E
	04369	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00367	04371	E
	04373	E
	04374	E
	04376	E
	04378	E
	04380	E
	04381	E
	04386	E
	04516	E
	04519	E
	04520	E
	04533	E
	04596	E
	04929	E
	04932	E
00367	04934	E
	04946	E
	04947	E
	04949	E
	04953	E
	04964	E
	04965	E
	04966	E
	04967	E
	04970	E
	04976	E
	04992	E
	04993	E
	05014	E
	05100	E
00367	05137	E
	05143	E
	05211	E
	08229	E
	08448	E
	08629	E
	08692	E
	09025	E
	09028	E
	09047	E
	09060	E
	09089	E
	12544	E
	12725	E
	12788	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00367	13152	E
	13488	R
	16421	E
	16821	E
	16884	E
	20517	E
	20917	E
	20980	E
	24613	E
	25013	E
	25076	E
	25426	E
	25427	E
	25428	E
	25429	E
	25431	E
00367	25432	E
	25433	E
	25436	E
	25437	E
	25438	E
	25439	E
	25440	E
	25441	E
	25442	E
	25444	E
	25445	E
	25450	E
	25467	E
	25473	E
	25479	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00367	25480	E
	25580	E
	25616	E
	25617	E
	25618	E
	25619	E
	25664	E
	25690	E
	25691	E
	29109	E
	29172	E
	29522	E
	29523	E
	29524	E
	29525	E
00367	29527	E
	29528	E
	29529	E
	29532	E
	29533	E
	29534	E
	29535	E
	29536	E
	29537	E
	29540	E
	29541	E
	29546	E
	29712	E
	29713	E
	29714	E
00367	29715	E
	29760	E
	32805	E
	33058	E
	33205	E
	33268	E
	33618	E
	33619	E
	33620	E
	33621	E
	33623	E
	33624	E
	33632	E
	33636	E
	33637	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00367	33665	E
	37301	E
	37719	E
	37728	E
	37732	E
	37761	E
	41397	E
	41460	E
	41824	E
	41828	E
	45493	E
	45556	E
	45920	E
	49589	E
	49652	E
00367	53748	E
	61696	E
	61697	E
	61698	E
	61699	E
	61710	E
	61711	E
	61712	E
00437	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00275	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00437	00424	R,E
	00500	R,E
	00737	R
	00775	R,E
	00813	R
	00819	R
	00833	R,E
	00836	E
	00838	R,E
	00850	R,E
	00852	R
	00855	R
	00857	R
	00858	R,E
	00860	R
00437	00861	R
	00862	R
	00863	R
	00865	R
	00866	R
	00869	R
	00870	R,E
	00871	R,E
	00874	R
	00875	E
	00875	R
	00880	R,E
	00897	R,E
	00903	R
	00905	R,E
00437	00912	R
	00914	R
	00915	R
	00916	R
	00920	R
	00921	R
	00922	R
	00923	R,E
	00924	R,E
	01025	R,E
	01026	R,E
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00437	01043	R,E
	01047	R,E
	01051	R
	01097	R,E
	01098	R
	01114	E
	01115	E
	01126	E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
00437	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01257	R
	01275	R
	01280	R
	01281	R
	01283	R
	04946	E
	05348	R,E
	28709	R,E
00720	00037	R
	00420	C
	00425	E
	00864	C
	01046	C
	01200 (13488)	R
	01208	R
	01256	C

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00737	00037	R
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	E
	00297	R
	00420	R
	00423	R,E
	00424	R
	00437	R
	00500	R,E
00737	00813	R,E
	00833	E
	00836	E
	00838	E
	00850	R
	00869	R,E
	00870	R
	00871	R
	00875	R,E
	00880	R
	00905	R
	01025	R
	01026	R
	01027	E
	01097	R
00737	01200 (13488)	R
	01208	R
	01252	R
	01253	R,E
	01280	R,E
	01287	R,E
	28709	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00775	00037	R
	00256	R,E
	00273	R
	00277	R,E
	00278	R
	00280	R,E
	00284	R
	00285	R,E
	00290	E
	00297	R,E
	00420	R
	00423	R,E
	00424	R
	00437	R,E
	00500	R,E
00775	00833	E
	00836	E
	00838	E
	00850	R
	00870	R,E
	00871	R
	00875	R,E
	00880	R
	00905	R,E
	01025	R
	01026	R,E
	01027	E
	01097	R,E
	01112	R
	01122	R
00775	01200 (13488)	R
	01208	R
	01252	R,E
	01257	R
	28709	E
00806	01137	E
	01200 (13488)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00808	00259	E
	00858	R,E
	00859	R,E
	00872	R,E
	00923	R,E
	00924	R,E
	01025	R,E
	01140	R,E
	01148	R,E
	01153	R,E
	01154	R,E
	01158	R
	01200 (17584)	R,L
	01208	R,L
	05347	R,E
	05348	R,E
00813	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R
	00500	R
	00737	R,E
	00819	R
	00838	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00813	00850	R
	00852	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R,E
	00870	R
	00871	R
	00874	R
	00875	R,L
	00880	R
	00897	R
	00903	R
	00912	R
00813	00916	R
	00920	R
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
	01043	R
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01253	R
	01280	R
	01287	R,E
	05349	R,E
00819	00037	R,L
	00256	R
	00273	R,L
	00274	R,E,L
	00275	R,E,L
	00277	R,L
	00278	R,L
	00280	R,L
	00281	R,E,L
	00282	R,E,L
	00284	R,L
	00285	R,L
	00290	E,L
	00297	R,L
	00367	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00819	00420	R
	00423	R
	00424	R
	00425	R,E
	00437	R
	00500	R,L
	00803	R,E
	00813	R
	00833	E
	00836	E
	00838	R
	00850	R,E
	00852	R
	00855	R
	00857	R
00819	00858	R,E
	00860	R
	00861	R
	00863	R
	00864	R
	00865	R
	00866	R
	00869	R
	00870	R
	00871	R,L
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
00819	00905	R
	00912	R
	00914	R
	00915	R
	00916	R
	00920	R
	00921	R
	00922	R
	00923	E
	00924	R,E
	01004	R
	01025	R
	01026	R
	01027	R,E,L
	01041	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00819	01042	R
	01043	R
	01047	R,L
	01051	R
	01088	R
	01089	R
	01097	R
	01098	R
	01112	R,E
	01114	R,E
	01122	R,E
	01123	R,E
	01126	E
	01130	R,E
	01132	R,E
00819	01137	E
	01140	R,E,L
	01141	R,E,L
	01142	R,E,L
	01143	R,E,L
	01144	R,E,L
	01145	R,E,L
	01146	R,E,L
	01147	R,E,L
	01148	R,E,L
	01149	R,E,L
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00819	01157	R,E
	01158	R,E
	01160	R,E
	01164	R,E
	01200 (13488)	R,L
	01208	R,L
	01250	R
	01251	R
	01252	R
	01253	R
	01254	R
	01255	R
	01257	R
	01258	R
	01275	R
	01280	R
	01281	R
	01283	R
	05348	R,E
00848	00924	R,E
	01123	R,E
	01148	R,E
	01154	R
	01158	R,E
	01200 (17584)	R,L
	01208	R,L
	05347	R,E
00849	00924	R,E
	01025	R,E
	01148	R,E
	01154	R,E
	01158	R
	01200 (17584)	R
	01208	R
	05347	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00850	00037	R,E,C,L
	00256	R,E
	00259	E
	00273	R,E,C,L
	00274	R,E
	00275	R,E
	00277	R,E,C,L
	00278	R,E,C,L
	00280	R,E,C,L
	00284	R,E,C,L
	00285	R,E,C,L
	00290	R,E
	00297	R,E,C,L
	00367	E
	00420	R
00850	00423	R
	00424	R,E
	00437	R,E
	00500	R,E,C,L
	00737	R
	00775	R
	00803	R,E
	00813	R
	00819	R,E
	00833	R,E
	00836	R,E
	00838	R,E
	00852	R
	00855	R
	00856	R
00850	00857	R
	00858	E
	00860	R
	00861	R
	00862	R
	00863	R
	00864	R
	00865	R
	00866	R
	00869	R
	00870	R,E
	00871	R,E,C,L
	00874	R
	00875	R
	00880	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00850	00897	R,E
	00903	R
	00905	R,E
	00912	R
	00914	R
	00915	R
	00916	R
	00920	R
	00921	R
	00922	R
	00923	R,E
	00924	R,E
	01004	R
	01025	R
00850	01026	R,E
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01043	R,E
	01047	R,C,L
	01051	R
	01088	R
	01089	R
	01097	R
	01098	R
	01100	R
	01112	R
00850	01114	R,E
	01122	R
	01126	E
	01130	R,E
	01132	R,E
	01140	R,E,L
	01141	R,E,L
	01142	R,E,L
	01143	R,E,L
	01144	R,E,L
	01145	R,E,L
	01146	R,E,L
	01147	R,E,L
	01148	R,E,L
	01149	R,E,L
	01153	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00850	01200 (13488)	R,L
	01208	R,L
	01250	R
	01251	R
	01252	R,E
	01253	R
	01254	R
	01255	R
	01256	R
	01257	R
	01275	R
	01280	R
	01281	R
	01283	R
	04953	E
	05348	R,E
00851	00259	E
	00367	E
	00423	R
	00500	R
	00875	R
	01200 (13488)	R
	01208	R
00852	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00852	00437	R
	00500	R,E
	00813	R
	00819	R
	00833	R,E
	00836	R
	00838	R,E
	00850	R
	00855	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R,E
00852	00871	R,E
	00874	R
	00875	R,E
	00880	R,E
	00897	R
	00903	R
	00905	R,E
	00912	R,E
	00916	R
	00920	R
	01025	R,E
	01026	R,E
	01027	R,E
	01040	R,E
00852	01041	R,E
	01042	R
	01043	R,E
	01047	R
	01088	R
	01097	R,E
	01153	R,E
	01200 (13488)	R,L
	01208	R,L
	01250	R
	01252	R
	01282	R
	05346	R,E
	28709	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00853	00367	E
	01200 (13488)	R
	01208	R
00855	00037	R
	00259	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R,E
	00297	R
	00367	E
	00437	R
	00500	R
	00819	R
	00833	R,E
00855	00836	R
	00850	R
	00852	R
	00857	R
	00866	E
	00870	R
	00871	R
	00878	R
	00880	R
	00912	R
	00915	R,E
	01025	R,E
	01026	R
	01027	R,E
	01040	R,E
00855	01041	R,E
	01042	R
	01043	R,E
	01088	R
	01200 (13488)	R,L
	01208	R,L
	01250	R
	01251	R
	01252	R
	01283	R
	05347	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00856	00259	E
	00273	E
	00367	E
	00424	R
	00500	R
	00803	R
	00850	R
	00862	R
	00916	R
	01200 (13488)	R,L
	01208	R,L
	01255	R
	05351	R,E
00857	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E
00857	00437	R
	00500	R,E
	00813	R
	00819	R
	00833	R,E
	00836	R
	00838	R,E
	00850	R
	00852	R
	00855	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00857	00871	R,E
	00874	R
	00875	R,E
	00880	R,E
	00897	R
	00903	R
	00905	R,E
	00912	R
	00916	R
	00920	R
	01025	R,E
	01026	R,E
	01027	R,E
	01040	R,E
00857	01041	R,E
	01042	R
	01043	R,E
	01088	R
	01097	R,E
	01200 (13488)	R
	01208	R
	01252	R
	01254	R
	01281	R
	01288	R,E
	05350	R,E
	28709	R,E
00858	00037	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E
	00808	R,E
	00819	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00858	00850	E
	00860	R,E
	00861	R,E
	00865	R,E
	00871	R,E
	00872	R,E
	00901	R,E
	00902	R,E
	00923	R,E
	00924	R,E
	01027	E
	01047	R,E,L
	01051	R,E
	01140	R,E,L
	01141	R,E,L
00858	01142	R,E,L
	01143	R,E,L
	01144	R,E,L
	01145	R,E,L
	01146	R,E,L
	01147	R,E,L
	01148	R,E,L
	01149	R,E,L
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00858	01164	R,E
	01200 (17584)	R
	01208	R
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05210	R,E
	05348	R,E
	08482	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	16804	R,E
	17248	R,E
00859	00808	R,E
	00872	R,E
	00901	R,E
	00902	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01164	R,E
	01200 (17584)	R
	01208	R
	04909	R,E
	04971	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	16804	R,E
	17248	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00860	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E
00860	00437	R
	00500	R,E
	00813	R
	00819	R
	00833	R,E
	00838	R,E
	00850	R
	00852	R
	00857	R
	00858	R,E
	00861	R
	00863	R
	00865	R
	00869	R
	00870	R,E
00860	00871	R,E
	00874	R
	00875	R,E
	00880	R,E
	00897	R
	00903	R
	00905	R,E
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E
	01025	R,E
	01026	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00860	01027	R,E
	01041	R
	01042	R
	01043	R
	01097	R,E
	01140	R,E
	01145	R,E
	01146	R,E
	01148	R,E
	01200 (13488)	R
	01208	R
	01252	R
	05348	R,E
	28709	R,E
00861	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E
00861	00437	R
	00500	R,E
	00813	R
	00819	R
	00833	R,E
	00838	R,E
	00850	R
	00852	R
	00857	R
	00858	R,E
	00860	R
	00863	R
	00869	R
	00870	R,E
	00871	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00861	00874	R
	00875	R,E
	00880	R,E
	00897	R
	00903	R
	00905	R,E
	00912	R
	00916	R
	00920	R
	00923	R,E
	00924	R,E
	01025	R,E
	01026	R,E
00861	01027	R,E
	01041	R
	01042	R
	01043	R
	01097	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	05348	R,E
	28709	R,E
00862	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00862	00437	R
	00500	R,E
	00803	R,E
	00833	R,E
	00838	R,E
	00850	R
	00856	R
	00870	R,E
	00871	R,E
	00875	R,E
	00880	R,E
	00905	R,E
	00916	R,E
	01025	R,E
	01026	R,E
00862	01027	R,E
	01097	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01255	R,E
	05351	R,E
	12712	R,E
	28709	R,E
00863	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00863	00437	R
	00500	R,E
	00813	R
	00819	R
	00833	R,E
	00838	R,E
	00850	R
	00852	R
	00857	R
	00860	R
	00861	R
	00865	R
	00869	R
	00870	R,E
	00871	R,E
00863	00874	R
	00875	R,E
	00880	R,E
	00897	R
	00903	R
	00905	R,E
	00912	R
	00916	R
	00920	R
	00923	R,E
	01025	R,E
	01026	R,E
	01027	R,E
	01041	R
	01042	R
00863	01043	R
	01051	R
	01097	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R
	01208	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00863	01252	R
	01275	R
	05348	R,E
	28709	R,E
00864	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E
00864	00425	C
	00500	R,E
	00720	C
	00819	R
	00833	R,E
	00838	R,E
	00850	R
	00870	R,E
	00871	R,E
	00875	R,E
	00880	R,E
	00905	R,E
	00918	R
	01008	R
	01025	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00864	01026	R,E
	01027	R,E
	01046	C
	01089	E,C
	01097	R,E
	01127	R
	01200 (13488)	R,C,L
	01208	R,C,L
	01252	R
	01256	E
	05352	E
	28709	R,E
00865	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00420	R,E
	00423	R,E
	00424	R,E
00865	00437	R
	00500	R,E
	00819	R
	00833	R,E
	00838	R,E
	00850	R
	00858	R,E
	00860	R
	00863	R
	00870	R,E
	00871	R,E
	00875	R,E
	00880	R,E
	00905	R,E
	00923	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00865	00924	R,E
	01025	R,E
	01026	R,E
	01027	R,E
	01097	R,E
	01142	R,E
	01143	R,E
	01148	R,E
	01200 (13488)	R
	01208	R
	01252	R
	05348	R,E
	28709	R,E
00866	00037	R
	00256	E,C
	00367	E
	00437	R
	00500	E
	00819	R
	00850	R
	00855	E
	00870	R
	00878	R
	00880	E
	00915	E
	01025	R,E
	01200 (13488)	R,L
	01208	R,L
	01251	R
	01252	R
	01283	R
	05347	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00867	00259	E
	00424	R
	00916	R,E
	01148	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	01200 (17584)	R,L
	01208	R,L
	04899	R,E
	04971	R,E
	05012	R,E
	05351	R,E
	09048	R,E
	12712	R,E
	16804	R,E
00868	00367	E
	00918	R
	01006	R
	01200 (13488)	R,E
	01208	R,E
00869	00037	R
	00256	R
	00259	E
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R
	00500	R,E
	00737	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00869	00813	R,E
	00819	R
	00838	R
	00850	R
	00852	R
	00857	R
	00860	R
	00861	R
	00863	R
	00870	R
	00871	R
	00874	R
	00875	R,E
	00880	R
	00897	R
00869	00903	R
	00912	R
	00916	R
	00920	R
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
	01043	R
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01253	R
	01254	R
	01280	R
	01287	R,E
	05349	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00872	00259	E
	00808	R,E
	00858	R,E
	00859	R,E
	00923	R,E
	00924	R,E
	01025	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
00872	01147	R,E
	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01200 (17584)	R,L
	01208	R,L
	05346	R,E
	05347	R,E
	05348	R,E
	09044	R,E
	09049	R,E
00874	00037	R,E
	00259	E
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R
	00813	R
	00819	R
	00838	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00874	00850	R
	00852	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R
	00871	R
	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
00874	00920	R
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
	01043	R
	01200 (13488)	E,L
	01208	E,L
	01252	E
	04970	E
00878	00855	R
	00866	R
	00880	R,E
	00915	R
	01025	R,E
	01131	R,E
	01200 (13488)	R
	01208	R
	01251	R
	01283	R,E
	05347	R,E
00891	00367	E
	00500	E
	00833	E
	01088	E
	01200(13488)	R

I

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00895	00290	E
	00367	E
	00500	E
	01027	E
	01041	E
	01200 (13488)	R,C,M
	01208	R,C,M
00896	00290	E
	00367	E
	01027	E
	01041	E
	01200 (13488)	R,C,M
	01208	R,C,M
00897	00037	R,E
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	E
	00297	R
	00367	E
	00423	R
	00437	R,E
	00500	E
	00813	R
	00819	R
00897	00838	R
	00850	R,E
	00852	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00903	R
	00912	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00897	00916	R
	00920	R
	01025	R
	01026	R
	01027	E
	01041	E
	01042	R
	01043	R
	01200 (13488)	R,C,M
	01208	R,C,M
	01252	E
00899	00259	E
00901	00037	R,E
	00259	E
	00500	R,E
	00858	R,E
	00859	R,E
	00902	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01148	R,E
	01156	R,E
	01157	R,E
	01200 (17584)	R,L
	01208	R,L
	05348	R,E
	05353	R,E
00902	00037	R,E
	00259	E
	00500	R,E
	00858	R,E
	00859	R,E
	00901	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01148	R,E
	01156	R,E
	01157	R,E
	01200 (17584)	R,L
	01208	R,L
	05348	R,E
	05353	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00903	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R
	00500	E
	00813	R
	00819	R
	00836	E
00903	00838	R
	00850	R
	00852	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00912	R
00903	00916	R
	00920	R
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
	01043	R
	01115	E
	01200 (13488)	R
	01208	R
	01252	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00904	00037	E
	00367	E
	00500	E
	01114	E
	01200 (13488)	R,L
	01208	R,L
00912	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R
	00500	R
	00813	R
	00819	R
	00838	R
00912	00850	R
	00852	R,E
	00855	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00912	00916	R
	00920	R
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
	01043	R
	01047	R
	01148	E
	01153	R,E
	01200 (13488)	R,L
	01208	R,L
	01250	R
	01252	R
	01282	R
	05346	R,E
00913	01200 (17584)	R
	01208	R
00914	00037	R
	00437	R
	00500	R
	00819	R
	00850	R
	01112	R,E
	01122	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01257	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00915	00037	R
	00259	R
	00367	E
	00437	R
	00500	R
	00819	R
	00850	R
	00855	R,E
	00866	E
	00870	R
	00878	R
	00880	R
	01025	R,E
	01131	R
	01154	R,E
	01167	R,E
	01200 (13488)	R,L
	01208	R,L
	01251	R
	01252	R
	01283	R
	05347	R,E
00916	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00424	R,E
	00437	R
	00500	R
	00803	R,E
	00813	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00916	00819	R
	00838	R
	00850	R
	00852	R
	00856	R
	00857	R
	00860	R
	00861	R
	00862	R,E
	00863	R
	00867	R,E
	00869	R
	00870	R
	00871	R
	00874	R
00916	00875	R
	00880	R
	00897	R
	00903	R
	00912	R
	00920	R
	01025	R
	01026	R
	01027	R
	01041	R
	01042	R
	01043	R
	01148	E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01255	R,E
	05351	R,E
	09048	R,E
	12712	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00920	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00423	R
	00437	R
	00500	R
	00813	R
	00819	R
	00838	R
00920	00850	R
	00852	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
	00880	R
	00897	R
	00903	R
	00905	R
00920	00912	R
	00916	R
	01025	R
	01026	R,L,M
	01148	E
	01155	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01254	R
	01281	R
	01288	R,E
	05350	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00921	00037	R
	00367	E
	00437	R
	00500	R
	00819	R
	00850	R
	00922	R
	01112	R,E
	01122	R
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01257	R,E
	05353	R,E
00922	00037	R
	00367	E
	00437	R
	00500	R
	00819	R
	00850	R
	00921	R
	01112	R
	01122	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01257	R,E
	05353	R,E
00923	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E
	00808	R,E
	00819	E
	00850	R,E
	00858	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00923	00860	R,E
	00861	R,E
	00863	R,E
	00865	R,E
	00871	R,E
	00872	R,E
	00901	R,E
	00902	R,E
	00924	R,L
	01043	R,E
	01047	R,E
	01051	R,E
	01140	R,E
	01141	R,E
	01142	R,E
00923	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01156	R,E
	01157	R,E
	01158	R,E
	01160	R,E
	01161	R,E
00923	01162	R,E
	01164	R,E
	01200 (17584)	R
	01208	R
	01252	R,E
	01275	R,E
	04909	R,E
	04971	R,E
	05210	R,E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
	16804	R,E
	17248	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00926	00834	E
	00951	E
	01200 (17584)	R
	01208	R
	01362	E
00927	00835	E
	00947	E
	01200 (13488)	E
	01208	E
00928	00837	E
	01200 (13488)	R
	01208	R
	01380	E
	01385	E
00941	00300	E
	00301	E
	01200 (13488)	E
	01208	E
	01351	E
00947	00835	E
	00927	E
	01200 (13488)	E
	01208	E
00951	00834	E
	00926	E
	00971	E
	01200 (13488)	R,E
	01208	R,E
	01362	E
	04930	E
00952	00300	E
	01200 (13488)	E
	01208	E
00953	00300	E
	01200 (17584)	E
	01208	E
00955	00300	E
	01200 (13488)	E
	01208	E
00960	01200 (17584)	E
	01208	E
00961	01200(13488)	R
00963	01200 (13488)	E
	01208	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
00971	00834	E
	00951	E
	01200 (13488)	R
	01208	R
	01362	E
01004	00367	E
	00500	R
	00819	R
	00850	R
	01200 (13488)	R
	01208	R
01006	00367	E
	00868	R
	00918	R
	01200 (13488)	R,E
	01208	R,E
01008	00367	E
	00420	R
	00864	R
	01200 (13488)	R
	01208	R
01009	00037	E
	00273	E
	00277	E
	00278	E
	00280	E
	00284	E
	00290	E
	00297	E
	00367	R
	00423	E
	00500	E
	00833	E
	00836	E
	00870	E
	00871	E
	00875	E
	00880	E
	01025	E
	01026	E
	01200 (13488)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01010	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01011	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01012	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01013	00367	E
	00500	E
	01140	E
	01200 (13488)	R
	01208	R
01014	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01015	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01016	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01017	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01018	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01019	00367	E
	00500	E
	01200 (13488)	R
	01208	R
01020	00367	E
	00500	E
	01200 (17584)	R
	01208	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01021	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01023	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01040	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00437	R,E
	00500	R,E
	00833	R,E
	00836	R
	00850	R,E
01040	00852	R,E
	00855	R,E
	00857	R,E
	00870	R,E
	00871	R,E
	01025	R,E
	01026	R,E
	01027	R,E
	01041	R,E
	01042	R
	01043	R,E
	01088	R,E
	01200 (13488)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01041	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00423	R
	00437	R,E
	00500	R,E
	00813	R
	00819	R,E
01041	00833	R,E
	00836	R
	00838	R
	00850	R,E
	00852	R,E
	00855	R,E
	00857	R,E
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R,E
	00871	R,E
	00874	R
	00875	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01041	00880	R
	00895	E
	00896	E
	00897	E
	00903	R
	00912	R
	00916	R
	01025	R,E
	01026	R,E
	01027	R,E
	01040	R,E
	01042	R
	01043	R,E
	01088	R
	01200 (13488)	R,C,M
	01208	R,C,M
	01252	E
01042	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R
	00297	R
	00367	E
	00423	R
	00437	R
	00500	R,E
	00813	R
	00819	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01042	00833	R
	00836	R,E
	00838	R
	00850	R
	00852	R
	00855	R
	00857	R
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R
	00871	R
	00874	R
	00875	R
01042	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	01025	R
	01026	R
	01027	R
	01040	R
	01041	R
	01043	R
	01088	R
	01200 (13488)	R
	01208	R
01043	00037	R,E
	00273	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	R,E
	00297	R,E
	00367	E
	00423	R
	00437	R,E
	00500	R,E
	00813	R
	00819	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01043	00833	R,E
	00836	R
	00838	R
	00850	R,E
	00852	R,E
	00855	R,E
	00857	R,E
	00860	R
	00861	R
	00863	R
	00869	R
	00870	R,E
	00871	R,E
	00874	R
	00875	R
01043	00880	R
	00897	R
	00903	R
	00912	R
	00916	R
	00923	R,E
	01025	R,E
	01026	R,E
	01027	R,E
	01040	R,E
	01041	R,E
	01042	R
	01088	R
	01114	E
	01200 (13488)	R
	01208	R
01046	00367	E
	00420	C
	00425	C
	00500	E
	00720	C
	00864	C
	01089	C
	01127	R
	01200 (13488)	R,L
	01208	R,L
	01256	E
	05352	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01051	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00437	R
	00500	R
	00819	R
	00850	R
	00858	R,E
	00863	R
01051	00871	R
	00923	R,E
	00924	R,E
	01025	R
	01097	R
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
01051	01200 (13488)	E
	01208	E
	01252	R
	01275	R
	05348	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01088	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00290	R
	00297	R
	00367	E
	00500	R,E
	00819	R
	00833	R,E
	00836	R
	00850	R
01088	00852	R
	00855	R
	00857	R
	00870	R
	00871	R
	00875	R
	00891	E
	01025	R
	01026	R
	01027	R
	01040	R,E
	01041	R
	01042	R
	01043	R
	01126	E
01088	01200 (13488)	R,C,L,M
	01208	R,C,L,M

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01089	00037	R,E
	00367	E
	00420	C
	00425	E
	00500	R,E
	00819	R
	00850	R
	00864	E,C
	01046	C
	01127	C
	01148	E
	01200 (13488)	R,L
	01208	R,L
	01256	E
	05352	E
	09238	C
01098	00259	R
	00367	E
	00420	R
	00437	R
	00819	R
	00850	R
	01097	R
	01200 (13488)	R,E
	01208	R,E
	01252	R
01100	00037	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00500	R
	00850	R
	01200 (17584)	R
	01208	R
01101	00367	E
	00500	E
	01200 (17584)	R
	01208	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01102	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01103	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01104	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01105	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01106	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01107	00367	E
	00500	E
	01200 (17584)	R
	01208	R
01114	00037	E
	00367	E
	00437	E
	00500	E
	00819	R,E
	00836	E
	00850	R,E
	00904	E
	01043	E
	01115	E
	01200 (13488)	R
	01208	R
	28709	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01115	00037	E
	00367	E
	00437	E
	00500	E
	00836	E
	00903	E
	01114	E
	01200 (13488)	R,L
	01208	R,L
01124	00037	R,E
	00367	E
	00500	R,E
	01123	R,E
	01125	R,E
	01158	E
	01168	R,E
	01200 (13488)	R,L
	01208	R,C
	01251	R,E
	01283	R
	05347	R,E
01125	00367	E
	00500	R,E
	01123	R,E
	01124	R,E
	01200 (13488)	R,L
	01208	R,L
	01251	R,E
	01283	R
	05347	R,E
01126	00037	E
	00367	E
	00437	E
	00500	E
	00819	E
	00833	E
	00850	E
	01088	E
	01200 (13488)	R,C,M
	01200 (17584)	R
	01208	R
	01252	E
	13121	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01127	00420	R
	00864	R
	01046	R
	01089	C
	01256	C
01129	00500	R,E
	01130	R,E
	01200 (13488)	R
	01208	R
	01258	R,E
	05354	R,E
01131	00037	R,E
	00367	E
	00500	R
	00878	R,E
	00915	R
	01025	R
	01200 (13488)	R
	01208	R
	01251	R
	01283	R
	05347	R,E
01133	00500	R,E
	01132	R,E
	01200 (13488)	E
	01208	E
01161	00259	E
	00838	R,E
	00858	R,E
	00859	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01161	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	01200 (17584)	R,L
	01208	R,L
	04909	R,E
	04971	R,E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E
01162	00259	E
	00858	R,E
	00859	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
01162	01148	R,E
	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	04909	R,E
	04971	R,E
	05348	R,E
	09044	R,E
	09049	R,E
	09061	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01163	00924	R,E
	01130	R,E
	01148	R,E
	01164	R,E
	01200 (17584)	R
	01208	R
	05354	R,E
01167	00915	R,E
	01025	R,E
	01200 (17584)	R
	01208	R
	01251	R,E
	05347	R,E
01168	01123	R,E
	01124	R,E
	01200 (17584)	R
	01208	R
	01251	R,E
	05347	R,E
01250	00037	R
	00259	E
	00273	R,E
	00367	E
	00500	R,E
	00819	R
	00850	R
	00852	R
	00855	R
	00870	R,E
	00912	R
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01282	R
	05346	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01251	00037	R
	00256	R,E
	00259	E
	00367	E
	00500	R,E
	00819	R
	00850	R
	00855	R
	00866	R
	00878	R
	00880	R,E
	00915	R
01251	01025	R,E
	01123	R,E
	01124	R,E
	01125	R,E
	01131	R
	01167	R,E
	01168	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01283	R
	05347	E
01252	00037	R,E
	00256	R,E
	00259	E
	00273	R,E
	00274	R,E
	00275	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	E
	00297	R,E
	00367	E
	00420	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01252	00423	R
	00424	R
	00437	R
	00500	R,E
	00737	R
	00775	R,E
	00803	R,E
	00813	R
	00819	R
	00833	E
	00836	E
	00838	E
	00850	R,E
	00852	R
	00855	R
01252	00857	R
	00858	R,E
	00860	R
	00861	R
	00862	R
	00863	R
	00864	R
	00865	R
	00866	R
	00869	R
	00870	R
	00871	R,E
	00874	E
	00875	R
	00880	R
01252	00897	E
	00903	E
	00905	R
	00912	R
	00914	R
	00915	R
	00916	R
	00920	R
	00921	R
	00922	R
	00923	R,E
	00924	R,E
	01025	R
	01026	R
	01027	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01252	01041	E
	01047	R,E
	01051	R
	01097	R
	01098	R
	01112	R,E
	01122	R,E
	01123	R,E
	01126	E
	01130	R,E
	01132	R,E
	01137	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
01252	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (13488)	R,L
	01208	R,L
	01250	R
	01251	R
	01254	R
	01255	R
	01257	R
	01275	R
	01280	R
	01281	R
01252	01283	R
	05348	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01253	00037	R
	00259	E
	00367	E
	00423	R,E
	00500	R,E
	00737	R,E
	00813	R
	00819	R
	00850	R
	00869	R
	00875	R,E
	01200 (13488)	R,L
	01208	R,L
	01280	R
	01287	R,E
	05349	E
01254	00037	R
	00259	E
	00367	E
	00500	R,E
	00819	R
	00850	R
	00857	R
	00869	R
	00905	R,E
	00920	R
	01026	R,E,L,M
	01047	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01281	R
	01288	R,E
	05350	E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01255	00037	R
	00259	E
	00367	E
	00424	R,E
	00500	R,E
	00803	R,E
	00819	R
	00850	R
	00856	R
	00862	R,E
	00916	R,E
	01200 (13488)	R,L
	01208	R,L
	01252	R
	01281	R
	05012	R,E
	05351	E
01256	00259	E
	00367	E
	00420	C
	00425	E
	00500	R,E
	00720	C
	00850	R
	00864	E
	01046	E
	01089	E
	01127	C
	01200 (13488)	R,L
	01208	R,L
	05352	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01257	00037	R
	00259	E
	00367	E
	00437	R
	00500	R
	00775	R
	00819	R
	00850	R
	00914	R
	00921	R,E
	00922	R,E
	01112	R,E
	01122	R,E
	01200 (13488)	R
	01208	R
	01252	R
	05353	E
01258	00037	R,E
	00259	E
	00500	R,E
	00819	R
	01129	R,E
	01130	R,E
	01200 (13488)	R
	01208	R
	05354	E
01275	00037	R
	00256	R
	00273	R
	00277	R
	00278	R
	00280	R
	00284	R
	00285	R
	00297	R
	00367	E
	00437	R
	00500	R
	00819	R
	00850	R
	00858	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01275	00863	R
	00871	R
	00923	R,E
	00924	R,E
	01051	R
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
01275	01200 (13488)	R
	01208	R
	01252	R
	05348	R,E
01276	00367	E
	01200 (13488)	R
	01208	R
01277	00367	E
	01200 (13488)	E
	01208	E
01280	00037	R
	00367	E
	00423	R
	00437	R
	00500	R
	00737	R,E
	00813	R
	00819	R
	00850	R
	00869	R
	00875	R
	01200 (13488)	R
	01208	R
	01252	R
	01253	R
	01287	R,E
	04971	R,E
	05349	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01281	00037	R
	00367	E
	00437	R
	00500	R
	00819	R
	00850	R
	00857	R
	00905	R
	00920	R
	01026	R
	01148	R,E
	01155	R,E
	01200 (13488)	R
	01208	R
	01252	R
	01254	R
	01255	R
	05350	R,E
01282	00367	E
	00500	R
	00852	R
	00870	R
	00912	R
	01153	R,E
	01200 (13488)	R
	01208	R
	01250	R
	05346	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01283	00037	R
	00367	E
	00437	R
	00500	R
	00819	R
	00850	R
	00855	R
	00866	R
	00878	R,E
	00880	R
	00915	R
	01025	R
	01123	R
	01124	R
	01125	R
	01131	R
	01154	R,E
	01200 (13488)	R
	01208	R
	01251	R
	01252	R
	05347	R,E
01284	01200 (13488)	R
	01208	R
	05346	R,E
01285	01200 (13488)	R
	01208	R
	05346	R,E
01284	01200 (13488)	R
	01208	R
01287	00737	R,E
	00813	R,E
	00869	R,E
	00875	R,E
	01253	R,E
	01280	R,E
	05349	R,E
01288	00857	R,E
	00920	R,E
	01026	R,E
	01254	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01351	00300	E
	00301	E
	00941	E
	01200 (13488)	R
	01208	R
01362	00834	E
	00926	E
	00951	E
	00971	E
	01200 (13488)	R
	01200 (17584)	R
	01208	R
	04930	E
01374	01200 (17584)	R
	01208	R
	61956	R
01380	00837	E
	00928	E
	01200 (13488)	R,E
	01208	R,E
	01385	E
	04933	E
	13125	E
01382	00837	E
	01200 (13488)	R,E
	01208	R,E
01385	00837	E
	00928	E
	01200 (13488)	R
	01200(17584)	R
	01208	R
	01380	E
	04933	E
	13125	E
01391	01200 (21680)	C
	01208	C
04398	01200 (13488)	R
	01208	R
04533	00367	E
	01200 (13488)	R
	01208	R
04904	01200 (17584)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
04909	00037	R,E
	00500	R,E
	00858	R,E
	00859	R,E
	00875	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
04909	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R
	01208	R
	04971	R,E,L
	05348	R,E
	05349	R,E
	09044	R,E
	09049	R,E
	09061	R,E
04944	01200 (17584)	R
	01208	R
04945	01200 (17584)	R
	01208	R
04946	00367	E
	00437	E
	01200 (13488)	R
	01208	R
04947	00367	E
	01200 (13488)	R
	01208	R
04948	01200 (13488)	R
	01208	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
04949	00367	E
	01200 (13488)	R
	01208	R
04951	01200 (13488)	R
	01208	R
04952	01200 (13488)	R
	01208	R
04953	00367	E
	00850	E
	01200 (13488)	R
	01208	R
04954	01200 (17584)	R
	01208	R
04955	01200 (17584)	R
	01208	R
04956	01200 (17584)	R
	01208	R
04957	01200 (17584)	R
	01208	R
04958	01200 (17584)	R
	01208	R
04959	01200 (17584)	R
	01208	R
04960	01200 (13488)	R
	01208	R
04961	01200 (17584)	R
	01208	R
04962	01200 (17584)	R
	01208	R
04963	01200 (17584)	R
	01208	R
04964	00367	E
	01200 (13488)	R,E
	01208	R,E
04965	00367	E
	01200 (13488)	R
	01208	R
04970	00367	E
	00874	E
	01200 (13488)	E
	01208	E
04992	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
04993	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
05012	00867	R,E
	01200 (13488)	R
	01208	R
	01255	R,E
	04899	R,E
	12712	R,E
05023	01200 (13488)	E
	01208	E
05043	01200 (13488)	E
	01208	E
05047	01200 (13488)	R
	01208	R
05048	01200 (13488)	E
	01208	E
05049	01200 (13488)	E
	01208	E
50564	01200 (17584)	R,E
	01208	R,E
05057	01200 (13488)	E
	01208	E
05100	00367	E
	01200 (13488)	R
	01208	R
05104	01200 (17584)	R
	01208	R
	16804	R,E
	17248	R
05123	01200(17584)	R,L
	01208	R,L
05137	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
05142	01200 (13488)	R
	01208	R
05210	00037	E
	00858	R,E
	00923	R,E
	01148	R,E
	01159	E
	01200 (17584)	R
	01208	R
	05348	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
05211	00367	E
	01200 (13488)	R
	01208	R
05346	00852	R,E
	00870	R,E
	00872	R,E
	00912	R,E
	01148	R,E
	01153	R,E
	01200 (17584)	R,L
	01208	R,L
	01250	E
	01282	R,E
	01284	R,E
	01285	R,E
	09044	R,E
05347	00808	R,E
	00848	R,E
	00849	R,E
	00855	R,E
	00866	R,E
	00872	R,E
	00878	R,E
	00880	R,E
	00915	R,E
	01025	R,E
	01123	R,E
	01124	R,E
05347	01125	R,E
	01131	R,E
	01148	R,E
	01154	R,E
	01158	R,E
	01167	R,E
	01168	R,E
	01200 (17584)	R,L
	01208	R,L
	01251	E
	01283	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
05348	00037	R,E
	00259	E
	00273	R,E
	00275	R,E
	00277	R,E
	00278	R,E
	00280	R,E
	00284	R,E
	00285	R,E
	00290	E
	00297	R,E
	00425	R,E
	00437	R,E
	00500	R,E
	00808	R,E
05348	00819	R,E
	00850	R,E
	00858	R,E
	00860	R,E
	00861	R,E
	00863	R,E
	00865	R,E
	00871	R,E
	00872	R,E
	00901	R,E
	00902	R,E
	00923	R,E
	00924	R,E
	01027	E
	01047	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
05348	01051	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
	01200 (17584)	R
	01208	R
	01153	R,E
	01154	R,E
	01155	R,E
05348	01156	R,E
	01157	R,E
	01158	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01164	R,E
	01252	E
	01275	R,E
	04909	R,E
	04971	R,E
	05123	E
	05210	E
	08482	E
	09044	R,E
	09049	R,E
05348	09061	R,E
	12712	R,E
	16804	R,E
	17248	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
05349	00813	R,E
	00869	R,E
	00875	R,E
	01148	R,E
	01200 (17584)	R
	01208	R
	01253	E
	01280	R,E
	01287	R,E
	04909	R,E
	04971	R,E
	09061	R,E
05350	00500	R
	00857	R,E
	00920	R,E
	01026	R,E
	01148	R,E
	01155	R,E
	01200 (17584)	R,L
	01208	R,L
	01254	E
	01281	R,E
	09049	R,E
	09061	R,E
05351	00424	R,E
	00856	R,E
	00862	R,E
	00867	R,E
	00916	R,E
	01148	R,E
	01200 (17584)	R,L
	01208	R,L
	01255	E
	04899	R,E
	09048	R,E
	12712	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
05352	00420	C
	00425	R,E
	00864	E
	01046	E
	01089	E
	01148	R,E
	01200 (17584)	R,L
	01208	R,L
	01256	E
	09238	E
	16804	C
	17248	E
05353	00901	R,E
	00902	R,E
	00921	R,E
	00922	R,E
	01112	R,E
	01122	R,E
	01148	R,E
	01156	R,E
	01157	R,E
	01200 (17584)	R
	01208	R
	01257	E
05354	01129	R,E
	01130	R,E
	01148	R,E
	01163	R,E
	01164	R,E
	01200 (17584)	R
	01208	R
	01258	E
05470	01200 (17584)	R
	01208	R
	61956	R
05476	01200 (13488)	R,E
	01208	R,E
05478	01200 (13488)	R
	01208	R
05487	01200 (17584)	C
	01208	C
08493	01200 (13488)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
08629	00367	E
	01200 (13488)	R
	01208	R
09044	00858	R,E
	00859	R,E
	00870	R,E
	00872	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
09044	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R,L
	01208	R,L
	04909	R,E
	04971	R,E
	05346	R,E
	05348	R,E
	09049	R,E
	09061	R,E
	12712	R,E
	16804	R,E
09047	00367	E
	01200 (13488)	R
	01208	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
09048	00424	R
	00867	R,E
	00916	R,E
	01148	R,E
	01200 (17584)	R
	01208	R
	04899	R,E
	05351	R,E
	12712	R,E
09049	00500	E
	00858	R,E
	00859	R,E
	00872	R,E
	00923	R,E
	00924	R,E
	01026	R
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
	01149	R,E
09049	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (17584)	R
	01208	R
	04909	R,E
	04971	R,E
	05348	R,E
	05350	R,E
	09044	R,E
	09061	R,E
	12712	R,E
	16804	R,E
09056	01200 (13488)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
09060	00367	E
	00423	R,E
	00858	R,E
	00859	R,E
	00875	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E
09060	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	01161	R,E
	01162	R,E
	01200 (13488)	R,E
	01208	R,E
	04909	R,E
	04971	R,E
	05348	R,E
	05349	R,E
	05350	R,E
	09044	R,E
	09049	R,E
09061	01200 (17584)	R,L
	01208	R,L
09064	01200 (17584)	R
	01208	R
09066	01200 (13488)	E
	01208	E
09088	01200 (13488)	R,E,C,M
	01208	R,E,C,M
09089	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
09139	01200 (13488)	E
	01208	E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
09144	01200 (13488)	R
	01208	R
09145	01200 (13488)	R,E
	01208	R,E
09163	01200 (13488)	R
	01208	R
09238	00420	E
	00924	R,E
	01089	C
	01148	R,E
	01200 (17584)	R,L
	01208	R,L
	05352	E
	16804	C
	17248	C
09306	01200 (17584)	R
	01208	R
09444	01200 (17584)	R,C
	01200 (21680)	C
	01208	C
	61956	R
09447	12712	R,E
	01200 (17584)	R
	01208	R
09448	01200(21680)	C
09449	01200 (17584)	R
	01208	R
09572	01200 (13488)	R,E
	01208	R,E
09574	01200 (13488)	R
	01208	R
09577	01200 (17584)	C
	01200 (21680)	C
	01208	C
12725	00367	E
	01200 (13488)	R
	01208	R
13140	01200 (17584)	R
	01208	R
13143	01200 (17584)	R
	01208	R
13145	01200 (17584)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
13152	00367	E
	01200 (13488)	R
	01208	R
13156	01200 (17584)	R
	01208	R
13157	01200 (17584)	R
	01208	R
13162	01200 (17584)	R
	01208	R
13184	01200 (13488)	R,M
	01208	R,M
13185	01200 (13488)	R,C,M
	01208	R,C,M
13235	01200 (13488)	E
	01208	E
13240	01200 (13488)	R
	01208	R
13241	01200 (13488)	R
	01200 (17584)	R
	01208	R
16821	00367	E
	01200 (13488)	R
	01208	R
17221	01200 (17584)	R
	01208	R
17240	01200 (17584)	R
	01208	R
17248	00420	R,E
	00858	R,E
	00859	R,E
	00923	R,E
	00924	R,E
	01140	R,E
	01141	R,E
	01142	R,E
	01143	R,E
	01144	R,E
	01145	R,E
	01146	R,E
	01147	R,E
	01148	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
17248	01149	R,E
	01153	R,E
	01154	R,E
	01155	R,E
	01160	R,E
	01200 (17584)	R,L
	01208	R,L
	04971	R,E
	05104	R
	05348	R,E
	05352	E
	09238	C
	12712	R,E
	16804	R,E
17331	01200 (13488)	E
	01208	E
17337	01200 (17584)	R
	01208	R
20917	00367	E
	01200 (13488)	R
	01208	R
21344	01200 (17584)	R
	01208	R
21427	09027	E
	01200 (17584)	E
	01208	E
21433	01200 (13488)	R
	01208	R
24877	01200 (13488)	R
	01208	R
25013	00367	E
	01200 (13488)	R
	01208	R
25426	00367	E
	01200 (13488)	R
	01208	R
25427	00367	E
	01200 (13488)	R
	01208	R
25428	00367	E
	01200 (13488)	R
	01208	R
25429	00367	E
	01200 (13488)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
25431	00367	E
	01200 (13488)	R
	01208	R
25432	00367	E
	01200 (13488)	R
	01208	R
25433	00367	E
	01200 (13488)	R
	01208	R
25436	00367	E
	01200 (13488)	R
	01208	R
25437	00367	E
	01200 (13488)	R
	01208	R
25438	00367	E
	01200 (13488)	R
	01208	R
25439	00367	E
	01200 (13488)	R
	01208	R
25440	00367	E
	01200 (13488)	R
	01208	R
25441	00367	E
	01200 (13488)	R
	01208	R
25442	00367	E
25444	00367	E
	01200 (13488)	R,E
	01208	R,E
25445	00367	E
	01200 (13488)	R,E
	01208	R,E
25450	00367	E
	01200 (13488)	E
	01208	R
25467	00367	E
	01200 (13488)	R
	01208	R
25473	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
25479	00367	E
	01200 (13488)	R
	01208	R
25480	00367	E
	01200 (13488)	R
	01208	R
25502	01200 (17584)	R
	01208	R
25503	01200 (13488)	E
	01208	E
25504	01200 (13488)	R
	01208	R
25527	01200 (13488)	R,E
	01208	R,E
25580	01200 (13488)	R
	01208	R
25616	00367	E
	01200 (13488)	R
	01208	R
25617	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
25618	00367	E
	01200 (13488)	R
	01208	R
25619	00367	E
	01200 (13488)	R
	01208	R
25664	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
25690	00367	E
	01200 (13488)	R
	01208	R
25691	00367	E
	01200 (13488)	R
	01208	R
29109	00367	E
	01200 (13488)	R
	01208	R
29522	00367	E
	01200 (13488)	R
	01208	R

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
29523	00367	E
	01200 (13488)	R
	01208	R
29524	00367	E
	01200 (13488)	R
	01208	R
29525	00367	E
	01200 (13488)	R
	01208	R
29527	00367	E
	01200 (13488)	R
	01208	R
29528	00367	E
	01200 (13488)	R
	01208	R
29529	00367	E
	01200 (13488)	R
	01208	R
29532	00367	E
	01200 (13488)	R
	01208	R
29533	00367	E
29534	00367	E
	01200 (13488)	R
	01208	R
29535	00367	E
	01200 (13488)	R
	01208	R
29536	00367	E
	01200 (13488)	R
	01208	R
29537	00367	E
	01200 (13488)	R
	01208	R
29540	00367	E
	01200 (13488)	R,E
	01208	R,E
29541	00367	E
	01200 (13488)	R
	01208	R
29546	00367	E
	01200 (13488)	E
	01208	E
29623	01200 (13488)	R,E
	01208	R,E

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
29712	00367	E
	01200 (13488)	R
	01208	R
29713	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
29714	00367	E
	01200 (13488)	R
	01208	R
29715	00367	E
	01200 (13488)	R
	01208	R
29760	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
32805	01200 (13488)	R
	01208	R
33205	00367	E
	01200 (13488)	R
	01208	R
33618	00367	E
	01200 (13488)	R
	01208	R
33619	00367	E
	01200 (13488)	R
	01208	R
33620	00367	E
	01200 (13488)	R
	01208	R
33621	00367	E
	01200 (13488)	R
	01208	R
33623	00367	E
	01200 (13488)	R
	01208	R
33624	00367	E
	01200 (13488)	R
	01208	R
33632	00367	E
	01200 (13488)	R
	01208	R
33636	00367	E
	01200 (13488)	R,E
	01208	R,E

ASCII Conversion Table

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
33637	00367	E
	01200 (13488)	R
	01208	R
33665	00367	E
	01200 (13488)	R,C,M
	01208	R,C,M
33722	01200 (13488)	L
37301	00367	E
	01200 (13488)	R
	01208	R
37719	00367	E
	01200 (13488)	R
	01208	R
37728	00367	E
	01200 (13488)	R
	01208	R
37732	00367	E
	01200 (13488)	R,E
	01208	R,E
37761	00367	E
	01200 (13488)	R,C
	01208	R,C
41397	00367	E
	01200 (13488)	R
	01208	R
41824	00367	E
	01200 (13488)	R
	01208	R
41828	00367	E
	01200 (13488)	R,E
	01208	R,E
45493	00367	E
	01200 (13488)	R
	01208	R
45920	00367	E
	01200 (13488)	R
	01208	R
49589	00367	E
	01200 (13488)	R
	01208	R
53685	01200 (17584)	R
	01208	R
54189	01200 (13488)	R
54289	01200 (13488)	R

Table 47. ASCII Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
61697	00367	E
	01200 (13488)	R
	01208	R
61698	00367	E
	01200 (13488)	R
	01208	R
61699	00367	E
	01200 (13488)	R
	01208	R
61700	01200 (13488)	R
	01208	R
61710	00367	E
	01200 (13488)	R
	01208	R
62337	01200 (13488)	R
62381	01200 (13488)	R

Unicode Conversion Table

This table defines the conversion techniques supported.

Table 48. Unicode Conversion Table

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	00037	E,L
	00256	E
	00259	E
	00273	E,L
	00275	E,L
	00277	E,L
	00278	E,L
	00280	E,L
	00282	E,L
	00284	E,L
	00285	E,L
	00290	E,C,L
	00293	E
	00297	E,L
	00300	E

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	00301	E
	00367	E,C
	00420	E,C,L
	00423	E
	00424	E,L
	00437	E,L
	00500	E,L
	00720	E
	00737	E
	00775	E
	00803	R
	00806	E
	00813	E,L
	00819	E,L
	00833	E,C,L
01200(13488)	00834	E
	00835	E
	00836	E,C,L
	00837	E
	00838	E,L
	00850	E,L
	00851	E
	00852	E,L
	00853	E
	00855	E,L
	00856	E,L
	00857	E
	00860	E
	00861	E,L
	00862	E,L
01200(13488)	00863	E
	00864	E,C,L
	00865	E
	00866	E,L
	00868	E
	00869	E,L
	00870	E,L
	00871	E,L
	00874	E,L
	00875	E,L
	00878	E
	00880	E,L
	00891	E,C
	00895	E,C,M
	00896	E,C,M

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	00897	E,C,M
	00903	E,C
	00904	E,C,L
	00905	E
	00912	E,L
	00914	E,L
	00915	E,L
	00916	E,L
	00918	E
	00920	E,L
	00921	E,L
	00922	E,L
	00923	L
	00927	E,L
	00928	E
	00941	E
01200(13488)	00947	E
	00951	E,L
	00952	E
	00955	E
	00961	E
	00963	E
	00971	E
	01004	E
	01006	E
	01008	E
	01009	E
	01010	E
	01011	E
	01012	E
	01013	E
	01014	E

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	01015	E
	01016	E
	01017	E
	01018	E
	01019	E
	01025	E,L
	01026	E,L
	01027	E,C,L
	01040	E,C
	01041	E,C,M
	01042	E,C
	01043	E,C
	01046	E,L
	01047	E,L
	01051	E
01200(13488)	01088	E,C,L,M
	01089	E,L
	01097	E
	01098	E
	01112	E,L
	01114	E,C
	01115	E,C,L
	01122	E,L
	01123	E,L
	01124	E,L
	01125	E,L
	01126	E,C,M
	01129	E
	01130	E
01200(13488)	01131	E
	01132	E
	01133	E
	01137	E
	01202	*
	01208	*
	01250	E,L
	01251	E,L
	01252	E,L
	01253	E,L
	01254	E,L
	01255	E,L
	01256	E,L
	01257	E
	01258	E
	01275	E

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	01276	E
	01277	E
	01280	E
	01281	E
	01282	E
	01283	E
	01284	E
	01285	E
	01351	E
	01362	E
	01380	E
	01382	E
	01385	E
	04133	E
	04369	E
01200(13488)	04371	E
	04373	E
	04374	E
	04376	E
	04378	E
	04380	E
	04381	E
	04386	E,C
	04393	E
	04396	E
	04397	E
	04516	E,C
	04519	E
	04520	E
	04533	E
01200(13488)	04596	E
	04929	E,C
	04930	E
	04931	E
	04932	E,C
	04933	E
	04934	E
	04946	E
	04947	E
	04948	E
	04949	E
	04951	E
	04952	E
	04953	E
	04960	E

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	04964	E
	04965	E
	04966	E
	04967	E
	04970	E
	04976	E
	04992	E,C,M
	04993	E,C,M
	05012	E
	05014	E
	05023	E
	05043	E
	05047	E
	05048	E
	05049	E
01200(13488)	05067	E
	05100	E
	05137	E,C,M
	05142	E
	05143	E
	05211	E,C
	05476	E
	05478	E
	08229	E,C
	08448	E
	08492	E
	08493	E
	08612	E
	08629	E
	08692	E
01200(13488)	09025	E,C
	09026	E
	09028	E,C
	09030	E
	09047	E
	09056	E
	09060	E
	09066	E
	09088	R,E,C,M
	09089	E,C,M
	09139	E
	09144	E
	09145	E
	09163	E
	09572	E

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	09574	E
	12544	E
	12588	E
	12725	E
	12788	E
	13124	E,C,L
	13125	E
	13152	E
	13184	C,M
	13185	C,M
	13235	E
	13240	E
	13241	E
	16421	E
	16684	E
01200(13488)	16821	E
	16884	E
	17331	E
	20517	E
	20917	E
	20980	E
	21314	E
	21317	E
	21433	E
	24613	E
	24877	E
	25013	E
	25076	E
	25426	E
	25427	E
01200(13488)	25428	E
	25429	E
	25431	E
	25432	E
	25433	E
	25436	E
	25437	E
	25438	E
	25439	E
	25440	E
	25441	E
	25442	E
	25444	E
	25445	E
	25450	E

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	25467	E,C
	25473	E,C,M
	25479	E,C
	25480	E,C
	25503	E
	25504	E
	25527	E
	25580	E
	25616	E,C
	25617	E,C,M
	25618	E,C
	25619	E,C
	25664	E,C,M
	25690	E,C
	25691	E,C
01200(13488)	28709	E,C
	29109	E
	29172	E
	29522	E
	29523	E
	29524	E
	29525	E
	29527	E
	29528	E
	29529	E
	29532	E
	29533	E
	29534	E
	29535	E
	29536	E
01200(13488)	29537	E
	29540	E
	29541	E
	29546	E
	29623	E
	29712	E,C
	29713	E,C,M
	29714	E,C
	29715	E,C
	29760	E,C,M
	32805	E
	33058	E,C
	33205	E
	33268	E
	33618	E

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(13488)	33619	E
	33620	E
	33621	E
	33623	E
	33624	E
	33632	E
	33636	E
	33637	E
	33665	E,C,M
	33722	L
	37301	E
	37719	E
	37728	E
	37732	E
	37761	E,C,M
	41397	E
01200(13488)	41460	E
	41824	E
	41828	E
	45493	E
	45556	E
	45920	E
	49589	E
	49652	E
	53748	E
	54189	E
	54289	E
	61696	E
	61697	E
	61698	E
	61699	E
	61700	E
	61710	E
	61711	E
	61712	E
	62337	E
	62381	E

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(17584)	00274	E,L
	00286	E
	00425	E,C,L
	00808	E,L
	00848	E,L
	00849	E
	00858	E
	00859	E
	00867	E,L
	00872	E,L
	00901	E,L
	00902	E,L
	00913	E
	00923	E
	00924	E,L
01200(17584)	00926	E
	00953	E
	00960	E
	01020	E
	01021	E
	01023	E
	01100	E
	01101	E
	01102	E
	01103	E
	01104	E
	01105	E
	01106	E
	01107	E
	01126	E
01200(17584)	01132	E
	01140	E,L
	01141	E,L
	01142	E,L
	01143	E,L
	01144	E,L
	01145	E,L
	01146	E,L
	01147	E,L
	01148	E,L
	01149	E,L
	01153	E,L
	01154	E,L
	01155	E,L
	01156	E,L

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(17584)	01157	E,L
	01158	E,L
	01159	E,C,L
	01160	E,L
	01161	E,L
	01163	E
	01164	E
	01165	E,L
	01166	E
	01167	E
	01168	E
	01208	R
	01362	E
	01374	E
	01385	E
	04370	E
	04396	E
01200(17584)	04899	E
	04904	E
	04909	E
	04930	E
	04933	E
	04944	E
	04945	E
	04954	E
	04955	E
	04956	E
	04957	E
	04958	E
	04959	E
	04961	E
	04962	E

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(17584)	04963	E
	04971	E,L
	05056	E
	05104	E
	05123	E,C,L
	05210	E,C
	05346	E,L
	05347	E,L
	05348	E
	05349	E
	05350	E,L
	05351	E,L
	05352	E,L
	05353	E
	05354	E
01200(17584)	05470	E
	05472	E
	05487	C
	08482	E,C,L
	09027	E
	09042	E
	09044	E,L
	09048	E
	09049	E
	09061	E,L
	09064	E
	09238	E,L
	09306	E
	09444	E
	09447	E
	09449	E
	09577	C

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01200(17584)	12712	E,L
	13121	E,C,L
	13140	E
	13143	E
	13145	E
	13156	E
	13157	E
	13162	E
	13241	E
	16684	E
	16804	E,L
	17221	E
	17240	E
	17248	E,L
	17337	E
01200(17584)	20780	E
	21344	E
	21427	E
	25502	E
	53685	E
01200(21680)	01208	C
	01391	C
	09444	C
	09577	C
01202	01200	*
	01208	*
01208	00037	
	00256	
	00259	
	00273	
	00274	
	00275	
	00277	
	00278	
	00280	
	00282	
	00284	
	00285	
	00286	
	00290	
	00293	

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	00297	
	00300	
	00301	
	00367	
	00420	
	00423	
	00424	
	00425	
	00437	
	00500	
	00720	
	00737	
	00775	
	00803	
	00806	
01208	00808	
	00813	
	00819	
	00833	
	00834	
	00835	
	00836	
	00837	
	00838	
	00848	
	00849	
	00850	
	00851	
	00852	
	00853	
01208	00855	
	00856	
	00857	
	00858	
	00859	
	00860	
	00861	
	00862	
	00863	
	00864	
	00865	
	00866	
	00867	
	00868	
	00869	

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	00870	
	00871	
	00872	
	00874	
	00875	
	00878	
	00880	
	00891	
	00895	
	00896	
	00897	
	00901	
	00902	
	00903	
	00904	
01208	00905	
	00912	
	00913	
	00914	
	00915	
	00916	
	00918	
	00920	
	00921	
	00922	
	00923	
	00924	
	00926	
	00927	
	00928	
01208	00941	
	00947	
	00951	
	00952	
	00953	
	00955	
	00960	
	00963	
	00971	
	01004	
	01006	
	01008	
	01009	
	01010	
	01011	

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	01012	
	01013	
	01014	
	01015	
	01016	
	01017	
	01018	
	01019	
	01020	
	01021	
	01023	
	01025	
	01026	
	01027	
	01040	
01208	01041	
	01042	
	01043	
	01046	
	01047	
	01051	
	01088	
	01089	
	01097	
	01098	
	01100	
	01101	
	01102	
	01103	
	01104	
01208	01105	
	01106	
	01107	
	01112	
	01114	
	01115	
	01122	
	01123	
	01124	
	01125	
	01126	
	01129	
	01130	
	01131	
	01132	

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	01133	
	01137	
	01140	
	01141	
	01142	
	01143	
	01144	
	01145	
	01146	
	01147	
	01148	
	01149	
	01153	
	01154	
	01155	
01208	01156	
	01157	
	01158	
	01159	
	01160	
	01161	
	01163	
	01164	
	01165	
	01166	
	01167	
	01168	
	01200	
	01202	*
	01250	
01208	01252	
	01253	
	01254	
	01255	
	01256	
	01257	
	01258	
	01275	
	01276	
	01277	
	01280	
	01281	
	01282	
	01283	
	01284	

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	01285	
	01351	
	01362	
	01374	
	01380	
	01382	
	01385	
	01391	
	04133	
	04369	
	04370	
	04371	
	04373	
	04374	
	04376	
	04378	
01208	04380	
	04381	
	04386	
	04393	
	04396	
	04397	
	04516	
	04519	
	04520	
	04533	
	04596	
	04899	
	04904	
	04909	
	04929	

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	04930	
	04931	
	04932	
	04933	
	04934	
	04944	
	04945	
	04946	
	04947	
	04948	
	04949	
	04951	
	04952	
	04953	
	04954	
01208	04955	
	04956	
	04957	
	04958	
	04959	
	04960	
	04961	
	04962	
	04963	
	04964	
	04965	
	04966	
	04967	
	04970	
	04971	
01208	04976	
	04992	
	04993	
	05012	
	05014	
	05023	
	05043	
	05047	
	05048	
	05049	
	05056	
	05067	
	05100	
	05104	
	05123	

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	05137	
	05142	
	05143	
	05210	
	05211	
	05346	
	05347	
	05348	
	05349	
	05350	
	05351	
	05352	
	05353	
	05354	
	05470	
	05476	
01208	05478	
	05487	
	08229	
	08448	
	08482	
	08492	
	08493	
	08612	
	08629	
	08692	
	09025	
	09026	
	09027	
	09028	
	09030	

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	09042	
	09044	
	09047	
	09048	
	09049	
	09056	
	09060	
	09061	
	09064	
	09066	
	09088	
	09089	
	09139	
	09144	
	09145	
01208	09163	
	09238	
	09306	
	09444	
	09447	
	09449	
	09572	
	09574	
	09577	
	12544	
	12588	
	12712	
	12725	
	12788	
	13121	
01208	13124	
	13125	
	13140	
	13143	
	13145	
	13152	
	13156	
	13157	
	13162	
	13184	
	13185	
	13235	
	13240	
	13241	
	13488	

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	16421	
	16684	
	16804	
	16821	
	16884	
	17221	
	17240	
	17248	
	17331	
	17337	
	17584	
	20517	
	20780	
	20917	
	20980	
01208	21314	
	21317	
	21344	
	21427	
	21433	
	21680	
	24613	
	24877	
	25013	
	25076	
	25426	
	25427	
	25428	
	25429	
	25431	
01208	25432	
	25433	
	25436	
	25437	
	25438	
	25439	
	25440	
	25441	
	25442	
	25444	
	25445	
	25450	
	25467	
	25473	
	25479	

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	25480	
	25502	
	25503	
	25504	
	25527	
	25580	
	25616	
	25617	
	25618	
	25619	
	25664	
	25690	
	25691	
	28709	
	29109	
01208	29172	
	29522	
	29523	
	29524	
	29525	
	29527	
	29528	
	29529	
	29532	
	29533	
	29534	
	29535	
	29536	
	29537	
	29540	
01208	29541	
	29546	
	29623	
	29712	
	29713	
	29714	
	29715	
	29760	
	32805	
	33058	
	33205	
	33268	
	33618	
	33619	
	33620	

Unicode Conversion Table

Table 48. Unicode Conversion Table (continued)

FROM-CCSID	TO-CCSID	Technique Supported
01208	33621	
	33623	
	33624	
	33632	
	33636	
	33637	
	33665	
	37301	
	37719	
	37728	
	37732	
	37761	
	41397	
	41460	
	41824	
01208	41828	
	45493	
	45556	
	45920	
	49589	
	49652	
	53685	
	53748	
	61696	
	61697	
	61698	
	61699	
	61700	
	61710	
	61711	
	61712	
61956	01374	E
	05470	E
	09444	E

Note: Conversions marked with * (asterisk) or blank in the technique supported column are performed algorithmically, so no technique is applicable.

Case Conversion tables

These tables are provided by IBM for case conversion service.

Table 49. Case conversion service based on the Unicode Standard 3.0.1.

Table name	Description	Size
CUNA301C	to Upper Normal	128K

Table 49. Case conversion service based on the Unicode Standard 3.0.1. (continued)

Table name	Description	Size
CUNA301D	to Lower Normal	128K
CUNA301E	to Upper Special	128K
CUNA301F	to lower Special	128K
CUNA301G	to Upper Locale	128K
CUNA301H	to lower Locale	128K
CUNA301I	Tittle stops table	128K
CUNA301J	To Title	128K
CUNA301Y	Special Casing file	32 K

Table 50. Case conversion service based on the Unicode Standard 3.2.0.

Table name	Description	Size
CUNA320C	to Upper Normal	128K
CUNA320D	to Lower Normal	128K
CUNA320E	to Upper Special	128K
CUNA320F	to lower Special	128K
CUNA320G	to Upper Locale	128K
CUNA320H	to lower Locale	128K
CUNA320I	Tittle stops table	128K
CUNA320J	To Title	128K
CUNA320S	to Upper Normal Surrogates	0.5K
CUNA320T	to lower Normal Surrogates	0.5K
CUNA320Y	Special Casing file	32 K

Table 51. Case conversion service based on the Unicode Standard 4.0.1.

Table name	Description	Size
CUNA401C	to Upper Normal	128K
CUNA401D	to Lower Normal	128K
CUNA401E	to Upper Special	128K
CUNA401F	to lower Special	128K
CUNA401G	to Upper Locale	128K
CUNA401H	to lower Locale	128K
CUNA401I	Tittle stops table	128K
CUNA401J	To Title	128K
CUNA401S	to Upper Normal Surrogates	0.5K
CUNA401T	to lower Normal Surrogates	0.5K
CUNA401Y	Special Casing file	32 K

Table 52. Case conversion service based on the Unicode Standard 4.1.0.

Table name	Description	Size
CUNA410C	to Upper Normal	128K

IBM-supplied tables

Table 52. Case conversion service based on the Unicode Standard 4.1.0. (continued)

Table name	Description	Size
CUNA410D	to Lower Normal	128K
CUNA410E	to Upper Special	128K
CUNA410F	to lower Special	128K
CUNA410G	to Upper Locale	128K
CUNA410H	to lower Locale	128K
CUNA410I	Tittle stops table	128K
CUNA410J	To Title	128K
CUNA410S	to Upper Normal Surrogates	0.5K
CUNA410T	to lower Normal Surrogates	0.5K
CUNA410Y	Special Casing file	32 K

Table 53. Case conversion service based on the Unicode Standard 5.0.0.

Table name	Description	Size
CUNA500C	to Upper Normal	128K
CUNA500D	to Lower Normal	128K
CUNA500E	to Upper Special	128K
CUNA500F	to lower Special	128K
CUNA500G	to Upper Locale	128K
CUNA500H	to lower Locale	128K
CUNA500I	Tittle stops table	128K
CUNA500J	To Title	128K
CUNA500S	to Upper Normal Surrogates	0.5K
CUNA500T	to lower Normal Surrogates	0.5K
CUNA500Y	Special Casing file	32 K

Normalization tables

These tables are provided by IBM for normalization service.

Table 54. Normalization service based on the Unicode Standard 3.0.1.

Table name	Description	Size
CUNNCACT	Canonical class stop	64K
CUNNCDST	Canonical decomposition stop	128K
CUNNKDST	Compatibility decomposition stop	128K
CUNNCOST	Composition stop	128K
CUNNCDTB	Canonical decomposition table	10.25K
CUNNKDTB	Compatibility decomposition table	34K
CUNNCOMT	Composition table	7.25K

Table 54. Normalization service based on the Unicode Standard 3.0.1. (continued)

Table name	Description	Size
CUNCCNZ	Canonical class non zero	64K

Table 55. Normalization service based on the Unicode Standard 3.2.0.

Table name	Description	Size
CUNN320A	Canonical Decomposition Table	10.25K
CUNN320B	Canonical Decomposition Stop Table	128K
CUNN320C	Compatibility Decomposition Table	35K
CUNN320D	Compatibility Decomposition Stop Table	128K
CUNN320E	Composition Table	7.25K
CUNN320F	Composition Stop Table	128K
CUNN320G	Canonical Class Table	64K
CUNN320H	Canonical Class Non Zero	128K
CUNN320I	Canonical Decomposition Table for supplementary code points	8.75K
CUNN320J	Compatibility Decomposition Table for supplementary code points	48K
CUNN320K	Composition Table for supplementary code points	8.75K
CUNN320L	Canonical Class Non Zero for supplementary code points	0.025K

Table 56. Normalization service based on the Unicode Standard 4.0.1.

Table name	Description	Size
CUNN401A	Canonical Decomposition Table	10.25K
CUNN401B	Canonical Decomposition Stop Table	128K
CUNN401C	Compatibility Decomposition Table	35K
CUNN401D	Compatibility Decomposition Stop Table	128K
CUNN401E	Composition Table	7.25K
CUNN401F	Composition Stop Table	128K
CUNN401G	Canonical Class Table	64K
CUNN401H	Canonical Class Non Zero	128K
CUNN401I	Canonical Decomposition Table for supplementary code points	8.75K

IBM-supplied tables

Table 56. Normalization service based on the Unicode Standard 4.0.1. (continued)

Table name	Description	Size
CUNN401J	Compatibility Decomposition Table for supplementary code points	48K
CUNN401K	Composition Table for supplementary code points	8.75K
CUNN401L	Canonical Class Non Zero for supplementary code points	0.025K

Table 57. Normalization service based on the Unicode Standard 4.1.0.

Table name	Description	Size
CUNN410A	Canonical Decomposition Table	10.25K
CUNN410B	Canonical Decomposition Stop Table	128K
CUNN410C	Compatibility Decomposition Table	35K
CUNN410D	Compatibility Decomposition Stop Table	128K
CUNN410E	Composition Table	7.25K
CUNN410F	Composition Stop Table	128K
CUNN410G	Canonical Class Table	64K
CUNN410H	Canonical Class Non Zero	128K
CUNN410I	Canonical Decomposition Table for supplementary code points	8.75K
CUNN410J	Compatibility Decomposition Table for supplementary code points	48K
CUNN410K	Composition Table for supplementary code points	8.75K
CUNN410L	Canonical Class Non Zero for supplementary code points	0.025K

Collation tables

These tables are provided by IBM for collation service.

Table 58. Collation service based on the Unicode Standard 3.0.1.

Table name	Description	Size
CUNOBACE	Collation element (main) table	256K
CUNOMIDX	Index table	64K
CUNOTHLA	Thai Lao table	64K
CUNOFCD	Fast canonical decomposition stop	64K

Table 58. Collation service based on the Unicode Standard 3.0.1. (continued)

Table name	Description	Size
CUNOFKD	Fast compatibility decomposition stop	64K
CUNOFCO	Fast composition stop	64K
CUNOCODA	Contraction data	0.5K
CUNOTIDX	Contraction index	12.25K
CUNOEXDA	Expansion data	10.25K
CUNOEXIN	Expansion index	128K

Table 59. Collation service based on the Unicode Standard 4.0.0.

Table name	Description	Size
CUNO400A	Collation Element Main Table	640K
CUNO400B	Expansion Index Table	192K
CUNO400C	Expansion Elements Table	517K
CUNO400D	Contractions Index Table	32K
CUNO400E	Contractions Elements Table	1K
CUNO400F	Main Index Table	64K
CUNO400G	Rearrangement Values	64K
CUNO400H	Fast Canonical Decomposition	64K
CUNO400I	Fast Compatibility Decomposition	64K
CUNO400J	Fast Composition	64K
CUNO400K	Surrogates Collation Element Main Table	0.25K
CUNO400L	Surrogates Expansion Elements Table	15K
CUNO400M	Surrogates Contractions Elements Table	0.25K
CUNO400N	Surrogates Main Index Table	625K
CUNO400O	Surrogates Fast Canonical Decomposition	1.75K
CUNO400P	Surrogates Fast Compatibility Decomposition	4.75K
CUNO400Q	Surrogates Fast Composition	0.25K

Table 60. Collation service based on the Unicode Standard 4.1.0.

Table name	Description	Size
CUNO410A	Collation Element Main Table	640K
CUNO410B	Expansion Index Table	192K
CUNO410C	Expansion Elements Table	521K
CUNO410D	Contractions Index Table	32K
CUNO410E	Contractions Elements Table	6K
CUNO410F	Main Index Table	64K

IBM-supplied tables

Table 60. Collation service based on the Unicode Standard 4.1.0. (continued)

Table name	Description	Size
CUNO410G	Rearrangement Values	64K
CUNO410H	Fast Canonical Decomposition	64K
CUNO410I	Fast Compatibility Decomposition	64K
CUNO410J	Fast Composition	64K
CUNO410K	Surrogates Collation Element Main Table	0.25K
CUNO410L	Surrogates Expansion Elements Table	15.5K
CUNO410M	Surrogates Contractions Elements Table	0.25K
CUNO410N	Surrogates Main Index Table	629K
CUNO410O	Surrogates Fast Canonical Decomposition	1.75K
CUNO410P	Surrogates Fast Compatibility Decomposition	4.75K
CUNO410Q	Surrogates Fast Composition	0.25K

Stringprep tables

These profiles are provided by IBM for stringprep service.

Table 61. Profiles provided for stringprep service

Profile name	Description	Size
CUNSTCIS	UTF8STR_CS_PREP	64K
CUNSTCSP	UTF8STR_CS_PREP	8K
CUNSTMX1	UTF8STR_MIXED_PREP	8.5K
CUNSTMX2	UTF8STR_MIXED_PREP	64K

Appendix C. Defining CCSIDs and conversion tables

The steps that must be followed to establish user-defined CCSIDs and conversion tables are as follows:

1. Define the required CCSIDs in the knowledge base.
 - a. Specify the entries using the assembler macro CUNAIKBG in a data set.
 - b. Assemble the source data set and re-link the knowledge base module CUNMIKBS.
2. Create the required conversion tables.
 - a. Create a character map of a conversion table by using job CUNJITG1.
 - b. Create the binary format of a conversion table by using job CUNJITG2.
3. Modify job CUNJIUTL.

Defining a CCSID in the knowledge base

IBM-supplies a knowledge base module CUNMIKBS that describes all CCSIDs shipped with z/OS support for Unicode. It is a non-executable load module stored in SYS1.LINKLIB and is maintained by PTF when new CCSIDs are introduced. User-defined CCSIDs can be added to this knowledge base using the assembler macro CUNAIKBG which is supplied in SYS1.MACLIB. CUNSIUKB is a sample SMP/E USERMOD to assist this process. It is supplied in SYS1.SAMPLIB.

To add a user-defined CCSID, follow the steps (a - b):

Specifying the entries

Specify the entries in a source data set using the assembler macro CUNAIKBG, which is supplied in SYS1.MACLIB. CUNAIKBG accepts the following parameters:

Syntax

► `[label] CUNAIKBG CCSID=ccsid, [simple definition | mixed definition]` ►

simple definition:

`[ES=es, SUFFIX=suffix, CCDEF=ccdef]`

ccdef:

`[(sp,sub,nl,lf,cr,eof)]`

mixed definition:

`[ES=es, SUBIDS=sub, ACRI=acri]`

Defining CCSIDs and conversion tables

sub:



acri:



Notes:

- 1 specify 2 to 8 sub-ccsids

Parameters

ccsid The value *ccsid* specifies the user-defined CCSID to be inserted into the knowledge base. *ccsid* is to be specified in decimal form. It is a unique five-digit number in the range 57344-61439 (this range is reserved for private use). *ccsid* is required.

es The value *es* specifies the encoding scheme identifier. It is a four-digit identifier in hexadecimal form. The following encoding schemes are supported:

- Simple:
 - SBCS: 1100, 2100, 3100, 4100, 4105, 4155, 5100, 5150, 5160, 6100, 8100
 - DBCS: 1200, 2200, 3200, 5200, 7200, 8200, 9200
 - TBCS: 5700
 - UTF8: 7807
 - QBCS: 2900
- Mixed:
 - EBCDIC MBCS: 1301
 - EUC MBCS: 4403
 - PC MBCS: 2300, 2305, 3300, 2A00
 - TCP/IP MBCS: 5404, 5409, 540A

For more information about encoding schemes see Character Data Representation Architecture Reference (Chapter 3, 'CDRA Identifiers'). Note that the value *es* determines which of the other operands are mandatory or forbidden. *es* is required.

suffix The value *suffix* specifies a two-character alphanumeric identifier to be used in constructing the conversion table name. See Table 62 on page 455. *suffix* is required for simple CCSIDs. It must not be specified for mixed CCSIDs.

ccdef The value *ccdef* specifies the control function definitions. They must be specified within parenthesis, separated by commas in the following order:

1. *sp* (space)
2. *sub* (substitute)
3. *nl* (new line)
4. *lf* (line feed)

5. *cr* (carriage return)
6. *eof* (end of line)

Those value are indices into the tables described in Appendix C ('Control Character Reference Tables') in Character Data Representation Architecture Reference . *ccdef* is required for simple CCSIDs. It must not be specified for mixed CCSIDs.

sub The value *sub* specifies the list of sub-CCSIDs within parenthesis. The number of sub-CCSIDs must be between two and eight. *sub* is required for mixed CCSIDs. It must not be specified for simple CCSIDs.

acri The value *acri* specifies the type of the 'additional coding-related required information' (ACRI). *acri* consists of a *type* and an *id*. The *type* can be:

- PC (ACRI information for PC MBCS)
- EUC (ACRI information for EUC MBCS)
- TCP (ACRI information for 2022 TCP/IP MBCS)

type must match the type of the encoding scheme.

The *id* is an index into the ACRI tables described in Character Data Representation Architecture Reference . (Appendix C, 'ACRI Reference Tables') .

acri is required for mixed CCSID except EBCDIC MBCS. It must not be specified for simple CCSIDs and EBCDIC MBCS.

Assembling the source data set

Because CUNMIKBS is an SMP/E managed load module, IBM recommends that you modify it by using a SMP/E USERMOD. CUNSIUKB is supplied as a sample in SYS1.SAMPLIB:

```

++USERMOD(UMOD001)
/*****
*
*   Licensed Materials - Property of IBM
*
*   5694-A01
*
*   (C) Copyright IBM Corp. 2002
*
*   Status = HUN7707
*
*****
*
* Sample USERMOD for building a user-defined knowledge base
*
*****
* CHANGE ACTIVITY
*
*****/
.
++VER(Z038) FMID(HUN12A0).
++JCLIN.
//LINK      EXEC LINKS
//          PARM='NCAL,MAP,LIST,LET,NOXREF,REUS',
//          N=,NAME=LINKLIB
//LINKLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
```

Defining CCSIDs and conversion tables

```
ORDER CUNMIKBS
ORDER USERKBS
ORDER CUNMIEOF
MODE  AMODE(31),RMODE(ANY)
INCLUDE LINKLIB(CUNMIKBS)
INCLUDE LINKLIB(USERKBS)
ENTRY CUNMIKBS
NAME CUNMIKBS(R)

++SRC(USERKBS) DISTLIB(SCUNJCL) DISTMOD(LINKLIB).
USERKBS  CSECT
USERKBS  AMODE 31
USERKBS  RMODE ANY
*
      CUNAIKBG CCSID=57344,ES=1100,SUFFIX=AA,CCDEF=(1,1,1,1,1,1)
      CUNAIKBG CCSID=57345,ES=1200,SUFFIX=BB,CCDEF=(2,2,2,2,2,2)
      CUNAIKBG CCSID=57346,ES=1301,SUBIDS=(57344,57345)
*
      END    USERKBS
/*
```

From this sample you can see how to generate knowledge base entries for a EBCDIC MBCS CCSID and its components. By performing an SMP/E RECEIVE and APPLY the source gets assembled and load module CUNMIKBS is re-linked, containing the user-defined knowledge base CSECT USERKBS.

Notes:

1. **Do not change the ORDER statements of the link step. CUNMIEOF must be the last CSECT in the load module.**
2. **Be sure that an SMP/E ACCEPT has been performed for FMID HUNI2A0 before installing the USERMOD. Otherwise you cannot restore the original CUNMIKBS by performing an SMP/E RESTORE.**

Creating a conversion table

To create your own conversion table you have to

- Build a human-readable character map.
- Convert the human-readable map into the binary format.

There are two methods for creating a character map of a user-defined conversion table:

1. You can use an existing conversion table and create a character map out of this. Then you make some changes to the map and finally you convert it back to the binary format, or
2. You can type in the complete character map with an editor of your choice and convert it into the binary format.

Method 1 is the 'normal' way of creating a user-defined table and is described in more detail below . Follow the same process if using method 2.

Building a human-readable character map

A utility is supplied with job CUNJITG1 to build a character map from an existing binary conversion table. The following is an example of a character map.

```
%
% Character map created on 11/09/2000 at 09:54:33
%                               by CUNMITG1 Version 2.8.0
%
%      Table source: CUNRAAEB
%      Conversion mode: SBCS-SBCS
```

```
% Sub-character: <7F>
%
% 00037      00850
% -----
% <00>      <00>
% <01>      <01>
% <02>      <02>
% <03>      <03>
% <04>      <DC>
...
```

Each code point that maps to a target character different from the substitution character is listed in the character map. The mapping can be changed by editing the values within the '<' and '>' signs. You can also add or delete lines from the character map. Do not change the lengths of the values. Those must match the length defined by the encoding scheme of the knowledge base entry. Each code point must be mapped in exactly one separate line.

Note that the '%' sign in the first column indicates a comment line. These comment lines contain information from the knowledge base. You can add, change or delete comment lines as desired. You also can add comments at the end of each mapping line.

The following is an example of job CUNJITG1. It shows how to create a character map from the IBM-supplied conversion table from CCSIDs 00037 to 00850 with roundtrip technique. According to Table 62 on page 455, member CUNRAAEB is selected from data set UNI.SCUNTB. The character map is created in member MAP0AAEB of data set UNI.CHARMAP:

```
//UNITG1 JOB (ACCOUNT),'UNICODE-INST',NOTIFY=&SYSUID,
// MSGCLASS=X,MSGLEVEL=(1,1),TIME=60,CLASS=A,
// REGION=0M
//CUNMITG1 EXEC PGM=CUNMITG1,PARM='00037,00850,R'
//TABIN DD DISP=SHR,DSN=UNI.SCUNTB
//CHAROUT DD DISP=SHR,DSN=UNI.CHARMAP(MAP0AAEB)
//SYSPRINT DD SYSOUT=*
```

Required parameters for job CUNJITG1 are:

- PARM='from-ccsid,to-ccsid,technique' on the EXEC card where
 - from-ccsid
 - is the source CCSID of the conversion.
 - to-ccsid
 - is the target CCSID of the conversion.
 - technique
 - is one specific technique character as described in "Character conversion" on page 19.

Note: Both from-ccsid and to-ccsid must be defined in the knowledge base. CCSID 1200 is not resolved as described in "Character conversion" on page 19. You have to specify a distinct UCS-2 CCSID instead of 1200. You must specify a distinct technique character. A technique-search-order is not supported here.

- //TABIN DD: Specifies the concatenation of partitioned data sets that hold the binary tables. These data sets must be in FB 256 format.
- //CHAROUT DD: Specifies the data set which holds the created character map. This must be a sequential data set in FB 80 format.

Defining CCSIDs and conversion tables

- //SYSPRINT DD: Specifies the data set to hold the messages issued by the utility.

You can now make your changes to the character map MAP0AAEB and then go on with step (b)

Converting the character map to binary format

Use job CUNJITG2 to convert the (changed) character map back to the binary format required as input to the image generator. The following is an example that shows how to create a user-defined conversion table from the character map MAP0AAEB created in step (a). According to Table 62 on page 455, member CUN0AAEB is created in the private data set UNI.USERTBL:

```
//UNITG2 JOB (ACCOUNT),'UNICODE-INST',NOTIFY=&SYSUID,
// MSGCLASS=X,MSGLEVEL=(1,1),TIME=60,CLASS=A,
// REGION=0M
//CUNMITG2 EXEC PGM=CUNMITG2,PARM='57344,00850,0'
//CHARIN DD DISP=SHR,DSN=UNI.CHARMAP(MAP0AAEB)
//TABOUT DD DISP=SHR,DSN=UNI.USERTBL
//SYSPRINT DD SYSOUT=*
```

Required parameters for job CUNJITG2 are:

- PARM='from-ccsid,to-ccsid,technique' on the EXEC card where
 - from-ccsid
is the source CCSID of the conversion
 - to-ccsid
is the target CCSID of the conversion
 - technique
is one specific technique character as described in “Character conversion” on page 19. Use the range 0-9 instead of IBM’s technique characters which are alphabetic. This avoids conflicts in the names of the conversion tables.

Note: Both from-ccsid and to-ccsid must be defined in the knowledge base. CCSID 1200 is not resolved as described in “Character conversion” on page 19. You have to specify a distinct UCS-2 CCSID instead of 1200. You must specify a distinct technique character. A technique-search-order is not supported here.

- //CHARIN DD: Specifies the sequential data set which holds the modified character map. This must be in FB 80 format. Note that columns 73 to 80 are ignored.
- //TABOUT DD: Specifies the partitioned data set that holds the generated binary table. This must be a single data set in FB 256 format.
- //SYSPRINT DD: Specifies the data set to hold the messages issued by the utility.

Note that the substitution character is assigned to each code point that is not explicitly listed in the character map.

Modifying job CUNJIUTL

A new conversion image can now be generated that contains user-defined conversions. To do this the job CUNJIUTL must be modified.

The image generator searches the //TABIN DD statement of job CUNJIUTL for the required conversion tables. Each table is identified by its member name, in the form CUNtaabb.

Table 62. Naming conventions of conversion tables

Part of CUNtaabb	Description
t	is the <i>technique</i> character in the range R, E, C, L, M, or 0 - 9.
aa	is the suffix from the knowledge base entry representing the user-defined FROM-CCSID. The range is alphanumeric.
bb	is the suffix from the knowledge base entry representing the user-defined TO-CCSID. The range is alphanumeric.

The partitioned data set SCUNTBL contains the IBM-supplied conversion tables. It is possible to add your own conversion tables. IBM recommends using a separate partitioned data set where you put in your conversion tables as members. The data set must be in FB 256 format.

Concatenate the partitioned data set that holds the user-defined tables to //TABIN DD. Specify the appropriate CONVERSION control statements in job CUNJIUTL. The following example of job CUNJIUTL shows the valid usage of the user-defined CCSID 57344 and the user-defined conversion table between CCSIDs 57344 and 850 created by the previous samples:

```
//UNIUTL JOB (ACCOUNT), 'UNICODE-INST', NOTIFY=&SYSUID,
// MSGCLASS=X, MSGLEVEL=(1,1), TIME=60, CLASS=A,
// REGION=0M
//CUNMIUTL EXEC PGM=CUNMIUTL
//SYSPRINT DD SYSOUT=*
/* SYSIMG must be a FB 80 dataset *****
//SYSIMG DD DISP=SHR, DSN=UNI.IMAGES(CUNIMG01)
//TABIN DD DISP=SHR, DSN=UNI.SCUNTBL
// DD DISP=SHR, DSN=UNI.USER_TBL
//SYSIN DD *

/*****
/* example of input statements */
/*****
CONVERSION 57344, /* src is user-defined 57344 */
               850, /* tgt is IBM-supplied 850 */
               0; /* 0 is user technique char */
/*
```


Appendix D. Unicode Return and Reason Codes

Programming Interface information

This chapter includes z/OS support for Unicode return and reason codes.

Return code meanings

Table 63. Classification of return codes

Hexadecimal Return Code	Name	Meaning
0	CUN_RC_OK	No error, successfully completed.
4	CUN_RC_WARN	Warning, see reason code for more information.
8	CUN_RC_USER_ERR	User error, action required. See reason code for more information.
0C	CUN_RC_ENV_ERR	Error caused by the environment, the request cannot be processed. See reason code for more information.
10	CUN_RC_SYS_ERR	System error, inconsistent state. See reason code for more information.

The following table identifies the hexadecimal return and reason codes and the name associated with each reason code.

Table 64. Return and reason codes from Unicode Services

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
0	0	Name: CUN_RS_OK Meaning: The operation was successful. Action: None.	All
4	1	Name: CUN_RS_TRG_EXH Meaning: The target buffer was exhausted before all characters in the source buffer were converted. Action: Call the service again with either a target buffer large enough to hold the complete result of the conversion or keep the result of the conversion just performed and repeat calling the service with the part of the source buffer that was not converted and concatenate the results of the various conversions.	Conversion
4	2	Name: CUN_RS_INV_HANDLE_NOSET Meaning: Conversion is terminated. The handle is invalid because a SET UNI command has changed the environment. Action: Clear the handle and make sure that the FROM-CCSID and TO-CCSID are specified in the parameter area. Then call the service again.	Conversion

Return and reason codes

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
4	3	Name: CUN_RS_INV_HANDLE_SET Meaning: Conversion is terminated. The handle is invalid because a SET UNI command is in process and will change the conversion environment. Action: Clear the handle and make sure that the FROM-CCSID and TO-CCSID are specified in the parameter area. Consider waiting until the SET UNI command completes before calling the service again. Otherwise the same error condition is returned.	Conversion
4	4	Name: CUN_RS_NO_HANDLE Meaning: Conversion is terminated. No handle can be obtained because a SET UNI command is in process and will change the conversion environment. Action: Clear the handle and make sure that the FROM-CCSID and TO-CCSID are specified in the parameter area. Consider waiting until the SET UNI command completes before calling the service again. Otherwise the same error condition is returned.	Conversion
4	6	Name: CUN_RS_SUB_ACT_TERM Meaning: A character was found in the source buffer which cannot be converted into a TO-CCSID character and the CUNBNPRM_Sub_Action flag specifies terminate with error. Action: 1. Check whether the input string is correct and whether the correct conversion tables are used. 2. Turn on the Sub_Action flag to replace the invalid character with the target substitution character and call the conversion service again.	Conversion
4	7	Name: CUN_RS_MBC_INCOMPLETE Meaning: An incomplete character was found in the source buffer. This error happens when not all bytes of a multi-byte character are found in the source buffer. For example, the incomplete character can be found at the end of the source buffer if only the first byte of a double-byte character fits into the buffer. Action: Check whether the input string is correct. Make sure that the missing bytes are in the source string.	Conversion
4	8	Name: CUN_RS_CONTINUATION Meaning: For character casing, the character condition of being FINAL or NON_FINAL in a word could not be determined, as the character was the last character in the source buffer but not the last character in the caller's source data. The character in question is not cased. Action: Next call to casing service needs to start with the uncased character of this call as the first source character.	Conversion

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
4	9	Name: CUN_RS_STAGE2_FAIL Meaning: An indirect character conversion, which first converts from the source CCSID into UCS-2 characters in a workarea and then in a second stage from the workarea to the target buffer, experienced an error during stage 2 conversion. As there is no correlation of the failing stage 2 character to a certain stage 1 character, we reset the source and target pointers and length values to the original caller's values. The workarea pointer and length values are updated to point to the character which failed conversion. Action: Check whether the input string and the parameter settings used are reasonable.	Conversion
4	0A	Name: CUN_RS_WRK_EXH Meaning: The work buffer was exhausted before all characters in the target buffer could be processed. Action: Call the service again with the new parameter value in the work buffer, where the work buffer size must be at least the same size as the target buffer.	Normalization
4	0B	Name: CUN_RS_SOURCE_LEN_ZERO Meaning: For collation, one or both of the source input parameters or both (CUNBOPRM_Src1_Buf_Len or CUNBOPRM_Src2_Buf_Len) has length zero. This is a completely valid operation when a comparison is needed. When a sort key needs to be generated, users will not be notified about zero lengths. Action: Avoid the call to collation if one of the source input parameters has length zero (if CUNBOPRM_SKey_Opt=OFF). Performance will be improved. Results will be the same.	Collation
4	0C	Name: CUN_RS_MAL_CHAR_ACT_TERM Meaning: A character was found in the source buffer which is not a valid source character and could not be converted. CUNBCPRM_Mal_Action specifies "terminate with error". Action: Check whether the input string is correct and the correct conversion tables were used. An incomplete character may be causing a range check to fail.	Conversion
4	0D	Name: CUN_RS_INVALID_COLL_DATA_VER Meaning: The specified Collation version is already loaded into the Unicode DataSpace. Action: Check whether the specified collation version is correct and recall the service.	Collation
4	0E	Name: CUN_RS_INVALID_ALTERNATE_VALUE Meaning: Invalid alternate value. When Collation API version is set to CUNBOPRM_Ver2 or CUN4BOPR_Ver2 (31 and 64 bit respectively) there are only two valid values. If the invalid value is entered, this RS is set and the default value is set. Action: Call the service again with a valid alternate value: • ALTERNATE_NON_IGNOREABLE • ALTERNATE_SHIFTED	Collation

Return and reason codes

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component																								
4	0F	Name: CUN_RS_INVALID_NORMALIZATION_VALUE Meaning: Invalid normalization value. When Collation API version is set to CUNBOPRM_Ver2 or CUN4BOPR_Ver2 (31 and 64 bit respectively), there are only two valid values. If invalid value is entered, this RS is set and default is value is set. Action: Call the service again with a valid normalization value: • NORMALIZATION_OFF • NORMALIZATION_ON	Collation																								
4	11	Name: CUN_RS_LOCALES_AND_UCR_ARE_EXCLUSIVE Meaning: CUNBOPRM_Locale/CUN4BOPR_Locale (31-bit and 64-bit respectively) and CUNBOPRM_Collation_Rules_File/ CUN4BOPR_Collation_Rules_File (31-bit and 64-bit respectively) are mutually exclusive. If this were the case then this RS is set and Locale info has the highest priority over User Collation Rules sets. Action: Call the service again with CUNBOPRM_Locales/ CUN4BPRM_Locales (31-bit and 64-bit respectively) information or CUNBOPRM_Collation_Rules_File/ CUN4BOPR_Collation_Rules_File Collation (31-bit and 64-bit respectively) rules information but not both.	Collation																								
8	1	Name: CUN_RS_PARM_VER Meaning: Wrong version of the parameter area used. Action: Use the correct parameter area version constant provided in the following interface definition file. <table><tr><th>z/OS Unicode service</th><th>31-bit</th><th>64-bit</th></tr><tr><td>Character conversion</td><td>CUNBCIDF</td><td>CUN4BCID</td></tr><tr><td>Case Conversion</td><td>CUNBAIDF</td><td>CUN4BAID</td></tr><tr><td>Normalization</td><td>CUNBNIDF</td><td>CUN4BNID</td></tr><tr><td>Collation</td><td>CUNBOIDE</td><td>CUN4BOID</td></tr><tr><td>BIDI</td><td>CUNBBIDF</td><td>CUN4BBID</td></tr><tr><td>StringPrep</td><td>CUNBPIDF</td><td>CUN4BPID</td></tr><tr><td>Conversion information service</td><td>CUNBIIDF</td><td>CUN4BIID</td></tr></table> When the service is called successfully, CUNBIPRM_Return_Code = 0 and CUNBIPRM_Reason_Code = 0.	z/OS Unicode service	31-bit	64-bit	Character conversion	CUNBCIDF	CUN4BCID	Case Conversion	CUNBAIDF	CUN4BAID	Normalization	CUNBNIDF	CUN4BNID	Collation	CUNBOIDE	CUN4BOID	BIDI	CUNBBIDF	CUN4BBID	StringPrep	CUNBPIDF	CUN4BPID	Conversion information service	CUNBIIDF	CUN4BIID	Conversion
z/OS Unicode service	31-bit	64-bit																									
Character conversion	CUNBCIDF	CUN4BCID																									
Case Conversion	CUNBAIDF	CUN4BAID																									
Normalization	CUNBNIDF	CUN4BNID																									
Collation	CUNBOIDE	CUN4BOID																									
BIDI	CUNBBIDF	CUN4BBID																									
StringPrep	CUNBPIDF	CUN4BPID																									
Conversion information service	CUNBIIDF	CUN4BIID																									
8	2	Name: CUN_RS_WRK_BUF_SMALL Meaning: The work buffer is not large enough to hold at least one character of the maximum width of characters as used with the work buffer in indirect conversions. Action: Call the service again using a work buffer of larger size.	Conversion, Normalization, Collation, StringPrep																								

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
8	3	Name: CUN_RS_CCSID_NOT_SUPP Meaning: The specified conversion is not supported in the current conversion image. Action: Verify that the FROM-CCSID, TO-CCSID, and technique-search-order parameters on the call to the conversion services specify a conversion that has been included in the currently active conversion image. The DISPLAY UNI command can be used by the system operator to display the available conversions. Have your system administrator update the conversion image to include the specified conversion or change the parameter specification as appropriate. For DB2 users only: messages CUN2047I, CUN3004I, or CUN2007E may be displayed.	All
8	4	Name: CUN_RS_CASE_NOT_SUPP Meaning: It can be one of the following meanings: - An unsupported case conversion type was specified. Action: Recall the service with the conversion type parameter set to a supported conversion type. - An invalid locale name was introduced by CUNBAPRM_Locale or CUN4BAPR_Locale (31 and 64-bit respectively). Action: Recall the service with a valid locale name (See Appendix G, "Case service and locale valid names," on page 479).	Conversion
8	5	Name: CUN_RS_SUBCODEPAGE Meaning: The subcodepage number supplied by the caller in the input parameter list is invalid. It is not in the range of numbers valid for the specified conversion. Action: Call the service again with a subcodepage number in the valid range. A value of binary zero will let the conversion start with the default codepage for this conversion.	Conversion
8	6	Name: CUN_RS_TRG_BUF_SMALL Meaning: The target buffer is not large enough to hold at least one character of the maximum width of characters as given by the TO-CCSID. For CASE, Normalization, StrigPrep, and BIDI Unicode Services, target buffer is not large enough to hold at least one UTF-16 BE character. For Collation Service, target buffer is not large enough to hold at least one UTF-16 BE as intermediate normalized string or target buffer is not large enough to hold at least one sort-key value. Action: Call the service again using a target buffer of adequate length.	All

Return and reason codes

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
8	7	Name: CUN_RS_DDA_BUF_SMALL It can be either of the following reasons: Meaning: The caller supplied a DDA buffer that is not large enough for the storage required by the conversion services. Action: Call the service again using the required DDA_Buf_Len as described by the following constant: - For 31-bit callers: - CUNBCPRM_DDA_Req for character conversion (in interface definition file CUNBCIDF) - CUNBAPRM_DDA_Req for case conversion (in interface definition file CUNBAIDF) - CUNBNPRM_DDA_Req for character conversion (in interface definition file CUNBNIDF) - CUNBOPRM_DDA_Req for case conversion (in interface definition file CUNBOIDF) - CUNBIPRM_DDA_Req for conversion information (in interface definition file CUNBIIDF) - For 64-bit callers: - CUN4BCPR_DDA_Req for character conversion (in interface definition file CUN4BCID) - CUN4BAPR_DDA_Req for case conversion (in interface definition file CUN4BAID) - CUN4BNPR_DDA_Req for character conversion (in interface definition file CUN4BNID) - CUN4BOPR_DDA_Req for case conversion (in interface definition file CUN4BOID) - CUN4BIPR_DDA_Req for conversion information (in interface definition file CUN4BIID) Meaning: Technique "B" (BIDI) was specified and the DDA value in CUNBCPRM_DDA_Buf_Len (31 bit) or CUN4BCPR_DDA_Buf_Len (64 bit) does not meet the technique "B" DDA requirements. Action: Call the service using CUNBCPRM_DDA_Req2 (31 bit) or CUN4BCPR_DDA_Req2 (64 bit) provided in the interface definition file CUNBCIDF (31 bit) or CUN4BCID (64-bit).	Character Conversion, CASE Conversion, Normalization, Collation

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
8	8	Name: CUN_RS_DDA_MIN_SMALL Meaning: The caller supplied a DDA buffer that is not large enough for the storage needed for the initial call to CUNMCNV, CUN4MNCV, CUNMNORM, CUN4MNOR, CUNMOCOL, or CUN4MCOL. Action: You can take one of the following actions: - For CUNMCNV and CUN4MNCV, call the service again using the required DDA_BUF_LEN returned in the handle field HUCCE_DDA_BUF_LEN. - For Normalization (CUNMNORM and CUN4MNOR - 31 and 64-bit respectively) and Collation (CUNMOCOL and CUN4MCOL - 31 and 64-bit respectively) Services, use the following constants provided in the interface definition files: 31-bit callers: - CUNBNPRM_DDA_Req for character conversion (in interface definition file CUNBNIDF) - CUNBOPRM_DDA_Req for case conversion (in interface definition file CUNBIDF) 64-bit callers: - CUN4BNPRM_DDA_Req for character conversion (in interface definition file CUN4BNID) - CUN4BOPRM_DDA_Req for case conversion (in interface definition file CUN4BID)	Character Conversion, Normalization, Collation
8	9	Name: CUN_RS_INV_NORM_TYPE Meaning: An unsupported normalization type was specified in normalization parameter area (CUNBOPRM). Action: Call the service again using a valid normalization type: CUNBNPRM_D=1, CUNBNPRM_C=2, CUNBNPRM_KD=3, CUNBNPRM_KC=4.	Normalization
8	0A	Name: CUN_RS_INV_COLL_LEVEL Meaning: An unsupported collation level was specified. Action: Use a valid collation level in IDF_CUNBIDF.	Collation
8	0B	Name: CUN_RS_NO_SERV_AVAILABLE Meaning: An unavailable service was called in the active image. Action: Use SET command to load an image with the service available.	Case Normalization Collation
8	0C	Name: CUN_RS_WRK_EXHAUSTED Meaning: The work buffer was exhausted before all the Unicode characters (source buffers) were represented in collation elements (weights – work buffers). Action: Call the service again with new parameter value in the work buffer.	Collation

Return and reason codes

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
8	0D	Name: CUN_RS_TARG_EXHAUSTED Meaning: The target buffer was exhausted before all collation elements (work buffers) were represented in a sort key (target buffers). Action: Call the service again with new parameter value n in the target buffer.	Collation
8	0E	Name: CUN_RS_REAL_EXHAUSTED Meaning: There is not enough real storage to dynamically store the tables in the image during the conversion request. Action: Increase the Realstorage value using: - REALSTORAGE keyword from the CUNUNI parmlib member - REALSTORAGE keyword from the SETUNI console command and call the service again. The target buffer was exhausted before all collation elements (work buffers) were represented in a sort key (target buffers).	All
8	10	Name: CUN_RS_PROFILE_NOT_FOUND Meaning: The specified profile was not found on the default or the user specified dataset. Action: Verify that the profile parameter on the call to the conversion services exists on the dataset or is loaded. The system operator can use the DISPLAY UNI command to display the available profiles.	Stringprep
8	11	Name: CUN_RS_UNASSIGNED_CODE_POINT Meaning: A character was found in the source buffer which is in the unassigned range. CUNBPPRM_UNASSIGNER = 1 specifies "terminate with error". Action: Check whether the input string is correct.	Stringprep
8	12	Name: CUN_RS_STRINGPREP_FAILED_AT Meaning: Stringprep service failed while running one of the steps on the profile. Action: Call the service again.	Stringprep
8	14	Name: CUN_RS_SRC_BUFF_LEN_ZERO Meaning: Source buffer length is 0. Action: Call the service again with new parameter value in the source buffer length.	Stringprep
8	15	Name: CUN_RS_SRC_BUFF_PTR_NULL Meaning: Source buffer pointer is NULL. Action: Call the service again with a valid source buffer pointer.	Stringprep
8	16	Name: CUN_RS_TRG_BUFF_PTR_NULL Meaning: Target buffer pointer is NULL. Action: Call the service again with a valid target buffer pointer.	Stringprep

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
8	17	Name: CUN_RS_INVALID_NORM_DATA_VER Meaning: Invalid Normalization data Version was introduced when trying to use the Normalization services. Action: Call the service again with a valid normalization data version (31/64-bit respectively): • CUNBNPRM_NONE/CUN4BNPR_NONE • CUNBNPRM_UNI301/CUN4BNPR_UNI301 • CUNBNPRM_UNI320/CUN4BNPR_UNI320 • CUNBNPRM_UNI401/CUN4BNPR_UNI401 • CUNBNPRM_UNI410/CUN4BNPR_UNI410	Normalization
8	18	Name: CUN_RS_INVALID_COLLATION_KEYWORD_VALUES Meaning: Invalid collation keyword values were introduced in CUN4BOPR_Collation_Keyword or CUNBOPRM_Collation_Keyword (31/64-bit respectively) collation parameter area field. Action: Specify a valid keyword value and call the service again. For further information see CUN4BOPR_Collation_Keyword or CUNBOPRM_Collation_Keyword (31/64-bit respectively) on collation parameter description section .	Collation
8	19	Name: CUN_RS_INVALID_UCA_VERSION Meaning: Invalid Unicode collation version on fields: CUN4BOPR_UCA_Ver, CUN4BOPR_Collation_Keyword or CUNBOPRM_UCA_Ver, CUNBOPRM_Collation_Keyword (31/64-bit respectively) Action: Call the service again with a valid UCA version (31/64-bit respectively): • CUNBOPRM_UCAempty/CUN4BOPR_UCAempty • CUNBOPRM_UCA301/CUN4BOPR_UCA301 • CUNBOPRM_UCA400R1/CUN4BOPR_UCA400R1 • CUNBOPRM_UCA410/CUN4BOPR_UCA410	Collation
8	1C	Name: CUN_RS_TRG_BUFF_LEN_ZERO Meaning: Target buffer length is 0. Action: Call the service again with new parameter value in the target buffer length.	Stringprep
8	1D	Name: CUN_RS_WRK_BUFF_LEN_ZERO Meaning: Work buffer length is 0. Action: Call the service again with new parameter value in the work buffer length.	Stringprep
8	1E	Name: CUN_RS_WRK_BUFF_PTR_NULL Meaning: Work buffer pointer is NULL. Action: Call the service again with a valid work buffer pointer.	Stringprep

Return and reason codes

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
8	1F	Name: CUN_RS_OVERLAYING_COLLATION_KEYWORD Meaning: Collation keyword values are overlaid (same collation keywords appear more than once at CUNBOPRM_COLLATION_KEYWORD/ CUN4BOPR_COLLATION_KEYWORD (31-bit and 64-bit respectively). Action: Remove collation keywords that appear more than once.	Collation
8	20	Name: CUN_RS_INVALID_CASEFIRST_VALUE Meaning: Invalid case first value. Action: Call the service again with a valid case first value: • CASEFIRST_OFF • CASEFIRST_UPPER • CASEFIRST_LOWER	Collation
8	21	Name: CUN_RS_INVALID_LOCALE_INPUT Meaning: Invalid locale input. Action: See Appendix F, "Locales support," on page 471 for valid locales support.	Collation
8	33	Name: CUN_RS_UNSUPPORTED_UNIVER_FOR_CASE Meaning: An unsupported Unicode version was specified for CASE conversion service. Action: Specify one of the following: • CUNBAPRM_UNI300 / CUN4BAPR_UNI300 • CUNBAPRM_UNI301 / CUN4BAPR_UNI301 • CUNBAPRM_UNI320 / CUN4BAPR_UNI320 • CUNBAPRM_UNI401 / CUN4BAPR_UNI401 • CUNBAPRM_UNI410 / CUN4BAPR_UNI410 • CUNBAPRM_UNI500 / CUN4BAPR_UNI500	Case
0C	1	Name: CUN_RS_NO_UNI_ENV Meaning: The conversion environment is not set up. Action: IPL is necessary to initialize the conversion environment.	All
0C	2	Name: CUN_RS_NO_CONVERSION Meaning: The conversion services are not available. Action: IPL is necessary to load the conversion services.	Conversion
0C	3	Name: CUN_RS_DYN_ACTION_FAILED Meaning: The dynamic action failed because either: • There is no primary storage available, or • Unicode can not release storage needed for dynamic loading of tables, or • There were abnormal operations on the dynamic Action: Contact your system operator to load conversion services via SET UNI command. If problems persist, refer to message CUN4026I for more details.	Infrastructure

Table 64. Return and reason codes from Unicode Services (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Name of reason code Meaning and Action	Component
10	1	Name: CUN_RS_INCONSISTENT_UCCB Meaning: The UCCB is in an inconsistent state. Action: IPL is necessary to recover.	Infrastructure
10	2	Name: CUN_RS_INCONSISTENT_UCCE Meaning: The UCCE is in an inconsistent state. Action: IPL is necessary to recover.	Infrastructure
10	3	Name: CUN_RS_INV_CONVERSION Meaning: The contents of UCCE_CONVERSION are invalid. Action: IPL is necessary to recover.	Conversion
10	4	Name: CUN_RS_INCONSISTENT_UCAE Meaning: The UCAE is in an inconsistent state. Action: IPL is necessary to recover.	Infrastructure
10	5	Name: CUN_RS_INCONSISTENT_TABLES Meaning: The tables used for case conversion have inconsistent content. Action: Run the image generator to create a new image with the appropriate case tables and issue the SET UNI command to activate it.	Conversion
10	6	Name: CUN_RS_INCONSISTENT_UCNE Meaning: The UCAE is in an inconsistent state. Action: IPL is necessary to recover.	Infrastructure
10	7	Name: CUN_RS_INCONSISTENT_UCOE Meaning: The UCOE is in an inconsistent state. Action: IPL is necessary to recover.	Infrastructure
10	1C	Name: CUN_RS_WA_NOT_ALIGNED Meaning: An internal work area for the TRxx simulation code is not aligned on a double word boundary. Action: This is an internal error. Call the IBM Support Center. IPL is necessary to recover.	Conversion
10	20	Name: CUN_RS_TABLE_NOT_ALIGNED Meaning: The conversion table is not aligned on a page boundary. Action: This is an internal error. Call the IBM Support Center. IPL is necessary to recover.	Conversion

Image generator for z/OS support for Unicode™ – return codes

Return Code	Meaning	Action
0	Successful completion	The image has been created without problem. Check the listing for what has been generated.

Image generator for z/OS support for Unicode™ – return codes

Return Code	Meaning	Action
4	Warnings issued	A duplicate statement has been ignored. Check the listing for the following messages: • CUN1027W • CUN1029W
8	User error	The input (JCL or control statements) is incorrect. Check the listing for the following messages: • CUN1003E • CUN1018E • CUN1019E • CUN1020E • CUN1021E • CUN1022E • CUN1023E • CUN1024E • CUN1025E • CUN1004E • CUN1006E • CUN1007E • CUN1008E • CUN1009E • CUN1010E • CUN1011E • CUN1012E • CUN1026E
0C	Environment error	An error occurred during the handling of a file or the work storage. Check the listing for the following messages: • CUN1013E
20	Error recovery has occurred	The error recovery routine of the file I/O module detected an ABEND situation. Check the job log and the system console for additional z/OS error messages.

End of Programming Interface information

Appendix E. System control offsets to callable services

Programming Interface information

An alternative to loading or link-editing the service stub is to include the system control offset to the callable service in the code. The following sample code can be used to replace the CALL statement in the samples provided.

Examples for 31-bit callers

The following example assumes that register one (R1) is set up with the address of the parameter area.

```
L    R15,16          CVT - common vector table
L    R15,544(R15)    CSRTABLE
L    R15,60(R15)     CSR slot
L    R15,offset(R15) Address of the service
BALR R14,15          Branch and link
```

List of offsets for 31-bit services

The following table shows the offsets for 31 bit services.

Table 65. Offsets for 31-bit callers.

Interface description	Decimal offset
Character conversion	172
Case conversion	180
Normalization	212
Collation	228

Examples for 64-bit callers

The following example assumes that register one (R1) is set up with the address of the parameter area.

```
LLGT R15,16          CVT - common vector table
L    R15,544(R15)    CSRTABLE
L    R15,60(R15)     CSR slot
L    R15,offset(R15) Address of the service
BASR R14,15          Branch
```

List of offsets for 64-bit services

The following table shows the offsets for 64-bit services.

Table 66. Offsets for 64-bit callers.

Interface description	Decimal offset
Character conversion	204
Case conversion	196
Normalization	220
Collation	236

End of Programming Interface information

Appendix F. Locales support

Programming Interface information

The following table lists the locales supported in the data set SYS1.SCUNLOCL.

Table 67. Locales support for CUNBOPRM_Collation_Keyword/
CUN4BOPR_Collation_Keyword (31/64-bit)

#	Parameter Value	Language	Region	Variant	Member Name
1	LAF	Afrikaans			CUNAF
2	LAF_RZA	Afrikaans	South Africa		CUNAFZA
3	LAM	Amharic			CUNAM
4	LAM_RET	Amharic	Ethiopia		CUNAMET
5	LAR	Arabic			CUNAR
6	LAR_RAE	Arabic	United Arab Emirates		CUNARAE
7	LAR_RBH	Arabic	Bahrain		CUNARBH
8	LAR_RDZ	Arabic	Algeria		CUNARDZ
9	LAR_REG	Arabic	Egypt		CUNAREG
10	LAR_RIN	Arabic	India		CUNARIN
11	LAR_RIQ	Arabic	Iraq		CUNARIQ
12	LAR_RJO	Arabic	Jordan		CUNARJO
13	LAR_RKW	Arabic	Kuwait		CUNARKW
14	LAR_RLB	Arabic	Lebanon		CUNARLB
15	LAR_RLY	Arabic	Libya		CUNARLY
16	LAR_RMA	Arabic	Morocco		CUNARMA
17	LAR_ROM	Arabic	Oman		CUNAROM
18	LAR_RQA	Arabic	Qatar		CUNARQA
19	LAR_RSA	Arabic	Saudi Arabia		CUNARSA
20	LAR_RSD	Arabic	Sudan		CUNARSD
21	LAR_RSY	Arabic	Syria		CUNARSY
22	LAR_RTN	Arabic	Tunisia		CUNARTN
23	LAR_RYE	Arabic	Yemen		CUNARYE
24	LBE	Belarusian			CUNBE
25	LBE_RBY	Belarusian	Belarus		CUNBEBY
26	LBG	Belarusian			CUNBG
27	LBG_RBG	Belarusian	Bulgaria		CUNBGBG
28	LBN	Bengali			CUNBN
29	LBN_RIN	Bengali	India		CUNBNIN
30	LCA	Catalan			CUNCA
31	LCA_RES	Catalan	Spain		CUNCAES
32	LCA_RES_VPREEURO	Catalan	Spain	Pre Euro support	CUNCAESP
33	LCS	Czech			CUNCS
34	LCS_RCZ	Czech	Czech Republic		CUNCSCZ
35	LDA	Danish			CUNDA
36	LDA_RDK	Danish	Denmark		CUNDADK
37	LDE	German			CUNDE

Describe the locales supported in the data set SYS1.SCUNLOCL.

Table 67. Locales support for CUNBOPRM_Collation_Keyword/
CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

#	Parameter Value	Language	Region	Variant	Member Name
38	LDE_RAT	German	Austria		CUNDEAT
39	LDE_RAT_VPREEURO	German	Austria	Pre Euro support	CUNDEATP
40	LDE_RBE	German	Belgin		CUNDEBE
41	LDE_RCH	German	Switzerland		CUNDECH
42	LDE_RDE	German	Germany		CUNDEDE
43	LDE_RDE_VPREEURO	German	Germany	Pre Euro support	CUNDEDEP
44	LDE_RLU	German	Luxembourg		CUNDELU
45	LDE_RLU_VPREEURO	German	Luxembourg	Pre Euro support	CUNDELUP
46	LDE_VPHONEBOOK	German		Telephone book	CUNDEH
47	LEL	English			CUNEL
48	LEL_RGR	English	Greece		
49	LEL_RGR_VPREEURO	English	Greece	Pre Euro support	CUNELGRP
50	LEN	English			CUNEN
51	LEN_RAU	English	Australia		CUNENAU
52	LEN_RBE	English	Belgium		CUNENBE
53	LEN_RBE_VPREEURO	English	Belgium	Pre Euro support	CUNENBEP
54	LEN_RBW	English	Botswana		CUNENBW
55	LEN_RCA	English	Canada		CUNENCA
56	LEN_RGB	English	Great Britain		CUNENGB
57	LEN_RGB_VPREEURO	English	Great Britain	Pre Euro support	CUNENGBP
58	LEN_RHK	English	Hong Kong S.A.R of China		CUNENHK
59	LEN_RIE	English	Ireland		CUNENIE
60	LEN_RIE_VPREEURO	English	Ireland	Pre Euro support	CUNENIEP
61	LEN_RIN	English	India		CUNENIN
62	LEN_RMT	English	Malta		CUNENMT
63	LEN_RNZ	English	New Zealand		CUNENNZ
64	LEN_RPH	English	Philippines		CUNENPH
65	LEN_RSG	English	Singapore		CUNENSG
66	LEN_RUS	English	United States of America		CUNENUS
67	LEN_RUS_VPOSIX	English	United States of America	Posix	CUNENUSX
68	LEN_RVI	English	Virgin Islands (USA)		CUNENVI
69	LEN_RZA	English	South Africa		CUNENZA
70	LEN_RZW	English	Zimbabwe		CUNENZW
71	LEO	Esperanto			CUNEO
72	LES	Spanish			CUNES
73	LES_RAR	Spanish	Argentina		CUNESAR

Describe the locales supported in the data set SYS1.SCUNLOCL.

Table 67. Locales support for CUNBOPRM_Collation_Keyword/
CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

#	Parameter Value	Language	Region	Variant	Member Name
74	LES_RBO	Spanish	Bolivia		CUNESBO
75	LES_RCL	Spanish	Chile		CUNESCL
76	LES_RCO	Spanish	Colombia		CUNESCO
77	LES_RCR	Spanish	Costa Rica		CUNESCR
78	LES_RDO	Spanish	Dominican Republic		CUNESDO
79	LES_REC	Spanish	Ecuador		CUNESEC
80	LES_RES	Spanish	Spain		CUNESSE
81	LES_RES_VPREEURO	Spanish	Spain		CUNESESP
82	LES_RGT	Spanish	Guatemala		CUNESGT
83	LES_RHN	Spanish	Honduras		CUNESHN
84	LES_RMX	Spanish	Mexico		CUNESMX
85	LES_RNI	Spanish	Nicaragua		CUNESNI
86	LES_RPA	Spanish	Panama		CUNESPA
87	LES_RPE	Spanish	Peru		CUNESPE
88	LES_RPR	Spanish	Puerto Rico		CUNESPR
89	LES_RPY	Spanish	Paraguay		CUNESPY
90	LES_RSV	Spanish	El Salvador		CUNESSV
91	LES_RUS	Spanish	United States of America		CUNESUS
92	LES_RUY	Spanish	Uruguay		CUNESUY
93	LES_RVE	Spanish	Venezuela		CUNESVE
94	LES_VTRADITIONAL	Spanish		Traditional Spanish sort.	CUNEST
95	LET	Estonian			CUNET
96	LET_REE	Estonian	Estonia		CUNETEE
97	LEU	Basque			CUNEU
98	LEU_RES	Basque	Spain		CUNEUES
99	LEU_RES_VPREEURO	Basque	Spain	Pre Euro support	CUNEUESP
100	LFA	Persian			CUNFA
102		Persian	Iran		CUNFAIR
103	LFI	Finnish			CUNFI
104	LFI_RFI	Finnish	Finland		CUNFIFI
105	LFI_RFI_VPREEURO	Finnish	Finland	Pre Euro support	CUNFIFIP
106	LFO	Faroese			CUNFO
107	LFO_RFO	Faroese	Faroe Islands		CUNFOFO
108	LFR	French			CUNFR
109	LFR_RBE	French	Belgium		CUNFRBE
110	LFR_RBE_VPREEURO	French	Belgium	Pre Euro support	CUNFRBEP
111	LFR_RCA	French	Canada		CUNFRCA
112	LFR_RCH	French	Switzerland		CUNFRCH
113	LFR_RFR	French	France		CUNFRFR

Describe the locales supported in the data set SYS1.SCUNLOCL.

Table 67. Locales support for CUNBOPRM_Collation_Keyword/
CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

#	Parameter Value	Language	Region	Variant	Member Name
114	LFR_RFR_VPREEURO	French	France	Pre Euro support	CUNFRFRP
115	LFR_RLU	French	Luxembourg		CUNFRLU
116	LFR_RLU_VPREEURO	French	Luxembourg		CUNFRLUP
117	LGA	Irish			CUNGA
118	LGA_RIE	Irish	Ireland		CUNGAIE
119	LGA_RIE_VPREEURO	Irish	Ireland	Pre Euro support	CUNGAIEP
120	LGL	Galician			CUNGL
121	LGL_RES	Galician	Spain		CUNGLES
122	LGL_RES_VPREEURO	Galician	Spain	Pre Euro support	CUNGLESP
123	LGU	Gujarati			CUNGU
124	LGU_RIN	Gujarati	India		CUNGUIN
125	LGV	Manx	Gaelic		CUNGV
126	LGV_RGB	Manx	Gaelic	Great Britain	CUNGVGB
127	LHE	Hebrew			CUNHE
128	LHE_RIL	Hebrew	Israel		CUNHEIL
129	LHI	Hindi			CUNHI
130	LHI_RIN	Hindi	India		CUNHIIN
131	LHI_VDIRECT	Hindi	Direct		CUNHID
132	LHR	Croatian			CUNHR
133	LHR_RHR	Croatian	Croatia		CUNHRHR
134	LHU	Hungarian			CUNHU
135	LHU_RHU	Hungarian	Hungary		CUNHUHU
136	LHY	Armenian			CUNHY
137	LHY_RAM	Armenian	Armenia		CUNHYAM
138	LHY_RAM_VREVISED	Armenian	Armenia	Revised	CUNHYAMR
139	LID	Indonesian			CUNID
140	LID_RID	Indonesian	Indonesia		CUNIDID
141	LIS	Icelandic			CUNIS
142	LIS_RIS	Icelandic	Iceland		CUNISIS
143	LIT	Italian			CUNIT
144	LIT_RCH	Italian	Switzerland		CUNITCH
145	LIT_RIT	Italian	Italy		CUNITIT
146	LIT_RIT_VPREEURO	Italian	Italy	Pre Euro support	CUNITITP
147	LIW	Hebrew			CUNIW
148	LIW_RIL	Hebrew	Israel		CUNIWIL
149	LJA	Japanese			CUNJA
150	LJA_RJP	Japanese	Japan		CUNJAJP
152	LKL	Greenlandic			CUNKL
153	LKL_RGL	Greenlandic	Greenland		CUNKLGL
154	LKN	Kannada			CUNKN
155	LKN_RIN	Kannada	India		CUNKNIN

Describe the locales supported in the data set SYS1.SCUNLOCL.

Table 67. Locales support for CUNBOPRM_Collation_Keyword/
CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

#	Parameter Value	Language	Region	Variant	Member Name
156	LKO	Korean			CUNKO
157	LKO_RKR	Korean	Korea		CUNKOKR
158	LK1	Konkani			CUNK1
159	LK1_RIN	KonKani	India		CUNK1IN
160	LKW	Cornish			CUNKW
161	LKW_RGB	Cornish	Great Britain		CUNKWGB
162	LLT	Lithuanian			CUNLT
163	LLT_RLT	Lithuanian	Lithuania		CUNLTLT
164	LLV	Latvian			CUNLV
165	LLV_RLV	Latvian	Latvia		CUNLVLV
166	LMK	Macedonian			CUNMK
167	LMK_RMK	Macedonian	Macedonia		CUNMKMK
168	LMR	Marathi			CUNMR
169	LMR_RIN	Marathi	India		CUNMRIN
170	LMT	Maltese			CUNMT
171	LMT_RMT	Maltese	Malta		CUNMTMT
172	LNB	Norwegian Bokmal			CUNNB
173	LNB_RNO	Norwegian Bokmal	Norway		CUNNBNO
174	LNL	Dutch			CUNNL
175	LNL_RBE	Dutch	Belgium		CUNNLBE
176	LNL_RBE_VPREEURO	Dutch	Belgium	Pre Euro support	CUNNLBEP
177	LNL_RNL	Dutch	the Netherlands		CUNNLNL
178	LNL_RNL_VPREEURO	Dutch	the Netherlands	Pre Euro support	CUNNLNLP
179	LNN	Norwegian Nynorsk			CUNNN
180	LNN_RNO	Norwegian Nynorsk	Norway		CUNNNNO
184	LOM	Oromo			CUNOM
185	LOM_RET	Oromo	Ethiopia		CUNOMET
186	LOM_RKE	Oromo	Kenya		CUNOMKE
187	LPL	Polish			CUNPL
188	LPL_RPL	Polish	Poland		CUNPLPL
189	LPT	Portuguese			CUNPT
190	LPT_RBR	Portuguese	Brazil		CUNPTBR
191	LPT_RPT	Portuguese	Portugal		CUNPTPT
192	LPT_RPT_VPREEURO	Portuguese	Portugal	Pre Euro support	CUNPTPTP
193	LRO	Romanian			CUNRO
194	LRO_RRO	Romanian	Romania		CUNRORO
195	LRU	Russian			CUNRU
196	LRU_RRU	Russian	Russia		CUNRURU
197	LRU_RUA	Russian	Ukraine		CUNRUUA

Describe the locales supported in the data set SYS1.SCUNLOCL.

Table 67. Locales support for CUNBOPRM_Collation_Keyword/
CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

#	Parameter Value	Language	Region	Variant	Member Name
198	LSH	Serbo-Croatian			CUNSH
199	LSH_RYU	Serbo-Croatian	Yugoslavia		CUNSHYU
200	LSK	Slovak			CUNSK
201	LSK_RSK	Slovak	Slovakia		CUNSKSK
202	LSL	Slovenian			CUNSL
203	LSL_RSI	Slovenian	Slovenia		CUNSLSI
204	LSO	Somali			CUNSO
205	LSO_RDJ	Somali	Djibouti		CUNSODJ
206	LSO_RET	Somali	Ethiopia		CUNSOET
207	LSO_RKE	Somali	Kenya		CUNSOKE
208	LSO_RSO	Somali	Somalia		CUNSOSO
209	LSQ	Albanian			CUNSQ
210	LSQ_RAL	Albanian	Albania		CUNSQAL
211	LSR	Serbian			CUNSR
212	LSR_RYU	Serbian	Yugoslavia		CUNSRYU
213	LSV	Swedish			CUNSV
214	LSV_RFI	Swedish	Finland		CUNSVFI
215	LSV_RSE	Swedish	Sweden		CUNSVSE
216	LSW	Swahili			CUNSW
217	LSW_RKE	Swahili	Kenya		CUNSWKE
218	LSW_RTZ	Swahili	Tanzania		CUNSWTZ
219	LTA	Tamil			CUNTA
220	LTA_RIN	Tamil	India		CUNTAIN
221	LTE	Telugu			CUNTE
222	LTE_RIN	Telugu	India		CUNTEIN
223	LTH	Thai			CUNTH
224	LTH_RTH	Thai	Tailand		CUNTHTH
226	LTI	Tigrinya			CUNTI
227	LTI_RER	Tigrinya	Eritrea		CUNTIER
228	LTI_RET	Tigrinya	Ethiopia		CUNTIET
229	LTR	Turkish			CUNTR
230	LTR_RTR	Turkish	Turkey		CUNTRTR
231	LUK	Ukrainian			CUNUK
232	LUK_RUA	Ukrainian	Ukrania		CUNUKUA
233	LVI	Vietnamese			CUNVI
234	LVI_RVN	Vietnamese	Vietnam		CUNVIVN
235	LZH	Chinese			CUNZH
236	LZH_RCN	Chinese	China		CUNZHCHN
237	LZH_RHK	Chinese	Hong Kong S.A.R of China		CUNZHKK
238	LZH_RMO	Chinese	Macao S.A.R of China		CUNZHMO
239	LZH_RSG	Chinese	Singapore		CUNZHSG
240	LZH_RTW	Chinese	Taiwan		CUNZHTW

Describe the locales supported in the data set SYS1.SCUNLOCL.

Table 67. Locales support for CUNBOPRM_Collation_Keyword/
CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

#	Parameter Value	Language	Region	Variant	Member Name
241	LZH_RTW_VSTROKE	Chinese	Taiwan	Stroke ordering	CUNZHTWS
242	LZH_VPINYIN	Chinese		Pin yin ordering	CUNZHY

End of Programming Interface information

Describe the locales supported in the data set SYS1.SCUNLOCL.

Appendix G. Case service and locale valid names

This section lists all the valid locale names for Case Service. You can specify those locale names at CUNBAPRM_Locale (31-bit) or CUN4BAPR_Locale (64-bit).

Table 68. Case service and locale valid names

Locale name	Language	Country/Region
Ar_AA	Arabic	Algeria, Bahrain, Egypt, Iraq, Jordan, Kuwait, Lebanon, Libya, Morocco, Oman, Qatar, Saudi Arabia, Syria, Tunisia, U.A.E., Yemen
az_AZ	Azeri	Azerbaijan
Be_BY	Byelorussian	Belarus
Bg_BG	Bulgarian	Bulgaria
Ca_ES	Catalan	Spain
Cs_CZ	Czech	Czech Republic
Da_DK	Danish	Denmark
De_AT	German	Austria
De_CH	German	Switzerland
De_DE	German	Germany
De_LU	German	Luxembourg
El_GR	Greek	Greece
En_AU	English	Australia
En_BE	English	Belgium
En_CA	English	Canada
En_GB	English	United Kingdom
En_HK	English	China (Hong Kong S.A.R.of China)
En_IE	English	Ireland
En_IN	English	India
En_JP	English	Japan
En_NZ	English	New Zealand
En_PH	English	Philippines
En_SG	English	Singapore
En_US	English	United States
En_ZA	English	South Africa
Es_AR	Spanish	Argentina
Es_BO	Spanish	Bolivia
Es_CL	Spanish	Chile
Es_CO	Spanish	Colombia
Es_CR	Spanish	Costa Rica
Es_DO	Spanish	Dominican Republic
Es_EC	Spanish	Ecuador

Table 68. Case service and locale valid names (continued)

Es_ES	Spanish	Spain
Es_GT	Spanish	Guatemala
Es_HN	Spanish	Honduras
Es_MX	Spanish	Mexico
Es_NI	Spanish	Nicaragua
Es_PA	Spanish	Panama
Es_PE	Spanish	Peru
Es_PR	Spanish	Puerto Rico
Es_PY	Spanish	Paraguay
Es_SV	Spanish	El Salvador
Es_US	Spanish	United States
Es_UY	Spanish	Uruguay
Es_VE	Spanish	Venezuela
Et_EE	Estonian	Estonia
Fi_FI	Finnish	Finland
Fr_BE	French	Belgium
Fr_CA	French	Canada
Fr_CH	French	Switzerland
Fr_FR	French	France
Fr_LU	French	Luxembourg
He_IL	Hebrew	Israel
Hr_HR	Croatian	Croatia
Hu_HU	Hungarian	Hungary
Id_ID	Indonesian	Indonesia
It_CH	Italian	Switzerland
Is_IS	Icelandic	Iceland
It_IT	Italian	Italy
Ja_JP	Japanese	Japan
Ko_KR	Korean	Korea
Iw_IL	Hebrew	Israel
Lt_LT	Lithuanian	Lithuania
Lv_LV	Latvian	Latvia
Mk_MK	Macedonian	Macedonia
Ms_MY	Malay	Malaysia
Nl_BE	Dutch	Belgium
Nl_NL	Dutch	The Netherlands
No_NO	Norwegian	Norway
Pl_PL	Polish	Poland
Pt_BR	Portuguese	Brazil
Pt_PT	Portuguese	Portugal
Ro_RO	Romanian	Romania

Table 68. Case service and locale valid names (continued)

Ru_RU	Russian	Russia
Sh_SP	Serbian (Latin)	Serbia
Sk_SK	Slovak	Slovakia
Sl_SI	Slovene	Slovenia
Sq_AL	Albanian	Albania
Sr_SP	Serbian (Cyrillic)	Serbia
Sv_SE	Swedish	Sweden
Th_TH	Thai	Thailand
*Tr_TR	Turkish	Turkey
UK_UA	Ukrainian	Ukraine
Zh_CN	Simplified Chinese	China (PRC)
Zh_TW	Traditional Chinese	Taiwan
Note: The Locale with an asterisk (*) in column one is the Locale supported in Unicode version 3.0.		

Appendix H. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

<http://www.ibm.com/systems/z/os/zos/bkserv/>

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Notices

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS™, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- Collation
- DB2
- DFSMS
- IBM
- IBMLink
- Language Environment
- MVS
- z/Architecture
- z/OS

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Glossary of terms and abbreviations

This glossary defines technical terms and abbreviations used in Unicode documentation. If you do not find the term you are looking for, view IBM Glossary of Computing Terms, located at: <http://www.ibm.com/ibm/terminology>

This glossary defines technical terms and abbreviations used in *z/OS Support for Unicode: Using Unicode Services*. If you do not find the term you are looking for, refer to the Index of this document or go to *IBM Glossary of Computing Terms* at

<http://www.ibm.com/terminology>

This glossary includes terms and definitions from:

American National Standard Dictionary for Information Systems, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036.

Character Data Representation Architecture Reference, copyright 1995 by International Business Machines Corporation. Copies can be purchased from IBM.

A

ACRI. additional coding-related information: A CDRA term referring to the additional information that is required to complete the definition associated with using particular encoding schemes. This information is in addition to the encoding scheme identifier, character set identifiers and code page identifiers that are associated with the case particular encoding scheme. An example for ACRI is the range of valid first bytes of double-byte code points in mixed single-byte and double-byte code.

ANSI. American National Standards Institute: The organization originally founded in 1918 to handle the problem of manufacturing interchangeable parts. Today ANSI does not develop standards but coordinates and accredits standards development in the United States of America.

ASCII. American National Standard Code for Information Interchange: The standard code, using a coded set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange between data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

B

big endian. Big endian is a format for the storage of binary data in which the most significant byte is placed first. Big endian is used by most hardware architectures including the z/Architecture. Also see the *little endian* entry in this glossary>.

binary comparison. Referenced in most cases as "collation". Compares two strings according to pre-set collation rules.

C

case conversion. Conversion of a lower case character to upper case and vice versa.

CCSID . coded character set identifier: A 16-bit number identifying a specific set of encoding scheme identifier, character set identifier(s), code page identifier(s), and additional coding related information, that uniquely identifies the coded graphic character representation used.

CDRA. character data representation architecture: An IBM architecture that defines a set of identifiers, resources, services, and conventions to achieve a consistent representation, processing, and interchange of graphic character data in mixed environments.

character. A member of a set of elements used for organization, control, or representation of data. A character can be a graphic character or a control character.

character conversion. Conversion between specified CCSIDs. The process of converting a set of characters from one CCSID to another CCSID.

character set. A defined set of characters. No coded representation is assumed.

code. A system of bit patterns to which a specific graphic or a control meaning has been assigned.

code page. A specification of code points from a defined encoding scheme for each graphic character in a set or in a collection of graphic character sets. Within a code page, a code point can have only one specific meaning. See also code point and encoding scheme.

code page conversion. The process of converting a set of characters from one CCSID to another CCSID. The term 'code page conversion' is not used in this documentation; instead the term 'character conversion' is used.

Glossary

code point. A unique bit pattern defined in a code. Depending on the code, a code point can be 7-bit, 8-bit, 16-bit, or other. Code points are assigned to a graphic character in a code page.

code set. See *coded character set*.

coded character . A control or graphic character with its assigned code point.

coded character set. A set of unambiguous rules that establish a character set and the one-to-one relationships between the characters of the set and their coded representations. (ISO/IEC)

collation level. Levels of cultural comparison that are taken into consideration when forming a sort key or performing a binary comparison of Unicode strings. See Chapter 7, “Collation” for more information.

collation rules. Rules which set the properties for Unicode strings. See Chapter 7, “Collation” for more information.

composite conversion. Converting a MBCS CCSID is performed by decomposing it into its individual CCSIDs and then converting the MBCS character stream by using the appropriate CCSIDs. This process is called ‘composite conversion’ (mixed CCSIDs are involved). Also see the *simple conversion* entry in this glossary.

control character.

1. (ISO/IEC 6429) A control function, the coded representation of which consists of a single bit combination.
2. A character whose occurrence in a particular context initiates, modifies, or stops a control function.

control function. (ISO/IEC 6429) An element of a character set that affects the recording, processing, transmission, or interpretation of data, and that has a coded representation of one or more bit combinations.

conversion image. The conversion services can only be used when conversion tables and control blocks are loaded into storage. Conversion tables and control blocks together are called ‘conversion image’ or simply ‘image’. The conversion image is created by the image generator which runs as a batch job.

conversion environment. When the conversion image is loaded into a common storage data space, the conversion environment is activated and the conversion services are ready to be used by callers.

conversion services. This document describes the conversion services that are offered by z/OS support for Unicode. Also see *character conversion* and *case conversion*.

CPGID. code page global identifier: A number between 00001 and 65534 that is assigned to identify a

code page. It may be expressed as a five-digit decimal number, a four-digit hexadecimal number, or a double-byte binary number.

D

DBCS. double-byte (coded) character set: A coded character set in which each character is represented by a double-byte code point. Some character sets, such as Kanji, are too rich in symbols to be able to represent all the characters using single-byte codes. A double-byte code character set is used to represent the symbols that make up such large character sets.

designator sequence. A sequence used by some ISO2022-based encodings for indicating the character sets to use when shifting characters are used. (Also see: *Lunde, Ken: Understanding CJKV Information Processing. Chinese, Japanese, Korean & Vietnamese Computing. 1999. ISBN: 1-56592-224-7, O'REILLY ASSOCIATES*)

direct conversion. When the conversion is performed in one step, it is called a direct conversion.

E

EBCDIC. IBM Extended Binary Coded Decimal Interchange Code: A coded character set consisting of 8-bit coded characters.

empty conversion environment. A conversion environment with no tables available for any service.

empty image. The image created as the result of an empty conversion environment.

encoding scheme. A set of specific definitions that describe the philosophy used to represent character data. The number of bits, the number of bytes, the allowable ranges of bytes, maximum number of characters, and meanings assigned to some generic and specific bit patterns, are some examples of specifications to be found in such a definition.

encoding scheme identifier. A 16-bit number assigned to uniquely identify a particular encoding scheme specification. See also encoding scheme.

endian. See the *big endian* and *little endian* entries in this glossary.

enforced subset. Tables that map only the matching characters between the source CCSID and the target CCSID. All other characters are replaced with a unique substitution character that indicates a substitution has occurred. Enforced subset tables should be used when the target datastream will be viewed or processed.

EUC. Extended Unix Code: an MBCS encoding that consists of up to four subcode pages.

F

FROM-CCSID. The CCSID you are converting from.

G

GB18030. Chinese standard that specifies an extended Codepage and a mapping table for conversion to and from Unicode DBCS. GB18030 is formed with 1,2 and 4 byte character sets. 1 and 2 byte parts are similar to UTF and are compatible with GBK encodings.

graphic character. (ISO 646-1983)

1. A character other than a control function that has a visual representation normally handwritten, printed, or displayed.
2. A character that can be displayed or printed.
3. A graphic symbol such as a numeric, alphabetic, or special character, or ideogram.

graphic character set. A defined set of graphic characters treated as an entity. No coded representation is assumed.

H

High-surrogate. A Unicode code value in the range U+D800 through U+DBFF.

I

IDF. interface definition file

image generator for z/OS support for Unicode. This is a batch job supplied by z/OS support for Unicode for creating a conversion image. The job sometimes is referred to as 'image generator'.

indirect conversion. When the conversion is performed using an intermediate CCSID, it is called an indirect conversion.

infrastructure. The infrastructure supplies all parts necessary to customize and establish the conversion services. It includes conversion tables and the commands SET UNI, SETUNI, and DISPLAY UNI.

intermediate CCSID. An indirect conversion uses an intermediate CCSID (CCSID-1200) to complete the conversion.

L

little endian. Little endian is a format for storage of binary data in which the least significant byte is placed first. Little endian is used by the Intel® hardware architectures. Also see the *big endian* entry in this glossary.

Low-surrogate. A Unicode code value in the range U+DC00 through U+DFFF.

lowercase. Pertaining to the small alphabetic characters, whether accented or not, as distinguished from the capital alphabetic characters. The concept of case also applies to alphabets such as Cyrillic and Greek, but not to Arabic, Hebrew, Thai, Japanese, Chinese, Korean, and many other scripts. Examples of lowercase letters are a, b, and c. Lowercase stands in contrast to uppercase.

M

MBCS. multi-byte character set: A set of characters in which each character is represented by 1 or more bytes.

mixed code page. It is a codepage specially defined to refer to a combination of SBCS and DBCS coded character sets (MBCS) that may be used in data streams or files. For example, CCSID 5035 is a mixed code page for Japanese that consists of Latin characters in CCSID 1027 and Kanji characters in CCSID 4396.

malformed character. Characters whose structure or range is not valid on the source code page, and therefore can not be converted. An example is an incomplete byte-string, thus misrepresenting a character and categorizing it as malformed.

N

normalization. The process of removing alternate representations of equivalent sequences from textual data to convert the data into a form which can be binary-compared for equivalence. In the Unicode Standard, normalization refers specifically to processing to ensure that canonically equivalent (and/or compatibility equivalent) strings have unique representations. For more information, refer to the Unicode Standard Annex #15, "Unicode Normalization Forms", and Chapter 6, "Normalization," on page 77.

normalization form. One of the four Unicode normalization forms defined in the Unicode Standard Annex #15, "Unicode Normalization Forms". See Chapter 6, "Normalization," on page 77 for more information.

normalization form C (NFC). The normalization form that results from the canonical decomposition of a Unicode string, followed by the replacement of all decomposed sequences by primary composites where possible. See to Chapter 6, "Normalization," on page 77 for more information.

Glossary

normalization form D (NFD). The normalization form that results from the canonical decomposition of a Unicode string. See Chapter 6, “Normalization,” on page 77 for more information.

normalization form KC (NFKC). The normalization form that results from the compatibility decomposition of a Unicode string, followed by the replacement of all decomposed sequences by primary composites where possible. See Chapter 6, “Normalization,” on page 77 for more information.

normalization form KD (NFKD). The normalization form that results from the compatibility decomposition of a Unicode string. See Chapter 6, “Normalization,” on page 77 for more information.

P

PC. personal computer: In the context of this document, it is the name for an extension of the ISO 646 (ANSI version) 7-bit code structure to an 8-bit structure.

Q

QBCS. quadruple-byte character set: A set of characters in which each character is represented by four bytes.

R

Round trip. Encoding that occurs when every code point in the source CCSID maps to a unique code point in the target CCSID. Using round trip tables ensure the capability of reversing the conversion, and recovering the complete original source datastream.

S

SBCS. single-byte character set: A set of characters in which each character is represented by one byte.

script. A collection of graphic symbols used for writing. A script is not related to either a language nor a country. Members of the same linguistic family can use different scripts. For example, the Latin script is used by most western European languages, while the Arabic script is used in Arabic countries as well as in Iran for Farsi and in Pakistan for Urdu.

simple code page. A codepage with a pure single-byte or pure double-byte encoding (SBCS, DBCS, and UCS-2).

simple conversion. A simple conversion is a conversion where no mixed CCSID is involved. Also see the *composite conversion* entry in this glossary.

sort key. A collation of weights determined by the collation level and collation rules. Also called sort key vector. See Sort key vector format for more information.

sub code page. A code page is called sub code page when it is mentioned in the context of the code page that make up a mixed codepage.

surrogate pair. A coded character representation for a single abstract character that consists of a sequence of two Unicode values, where the first value of the pair is a high-surrogate and the second is a low-surrogate.

T

TBCS. triple-byte character set: A set of characters in which each character is represented by three bytes.

technique. There may be multiple conversion tables available for converting one CCSID to another. The difference between conversion tables are the different techniques (for example, 'Round Trip'(R) or 'Enforced Subset'(E).

TO-CCSID. The CCSID you are converting to.

U

UCAE. Unicode case conversion control entry: Each UCAE contains control information for one kind of case conversion.

UCCB. Unicode conversion control block.

UCCE. Unicode character conversion control entry: Each UCCE contains control information for one kind of character conversion.

UCS. Abbreviation for **universal character set**, which is specified by International Standard ISO/IEC 10646.

UCS-2. ISO/IEC 10646 encoding form: universal character set coded in 2 octets.

Unicode Standard. A universal character encoding standard that supports the interchange, processing, and display of text that is written in any of the languages of the modern world. It can also support many classical and historical texts and is continually being expanded. The Unicode Standard is compatible with ISO/IEC 10646.

uppercase. Pertaining to the capital alphabetic characters, whether accented or not, as distinguished from the small alphabetic characters. The concept of case also applies to alphabets such as Cyrillic and Greek, but not to Arabic, Hebrew, Thai, Japanese, Chinese, Korean, and many other scripts. Examples of capital letters are A, B, and C. Uppercase stands in contrast to lowercase.

UTF-8. Unicode transformation format or UCS transformation format: 8-bit encoding form. The UTF-8 is the Unicode transformation format that serializes a Unicode scalar value as a sequence of one to four bytes.

UTF-16. Unicode transformation format or UCS transformation format: 16-bit encoding form. The UTF-16 is the Unicode transformation format that serializes a Unicode value as a sequence of two bytes, in either big endian or little endian format.

UTF-32. Unicode transformation format or UCS transformation format: 32-bit encoding form. The UTF-32 is the Unicode transformation format that serializes a Unicode value as a sequence of four bytes, in either big endian or little endian format.

W

Weight. A value that identifies each part of the collation level for each Unicode character. The values can be found at: <http://www.unicode.org/unicode/reports/tr10/allkeys.txt>

Index

Special characters

technique

definition in glossary 492

Numerics

0 – 9: User-defined conversions, value for
technique-character in CONVERSION 20

A

accessibility 483

ACRI

definition in glossary 489

additional coding-related information

see 'ACRI' definition in glossary 489

address space, primary

restriction while calling the case conversion

services 60

restriction while calling the character conversion

services 29

restriction while calling the conversion information

service 178

ALET

specified for character conversion 28

American National Standard Code for Information
Interchange

see 'ASCII' definition in glossary 489

American National Standards Institute

see 'ANSI' definition in glossary 489

amode

restriction while calling the case conversion

services 60

restriction while calling the character conversion

services 29

restriction while calling the conversion information

service 178

ANSI

definition in glossary 489

AR mode

restriction while calling the case conversion

services 60

restriction while calling the character conversion

services 29

restriction while calling the conversion information

service 178

ASC mode

restriction while calling the case conversion

services 60

restriction while calling the character conversion

services 29

restriction while calling the conversion information

service 178

ASCII

definition in glossary 489

authorization

restriction while calling the case conversion

services 60

restriction while calling the character conversion

services 29

restriction while calling the conversion information

service 177

B

Bidi

general description 5

big endian 4

character conversion 25

definition in glossary 489

C

C interface

mapping of parameters for case conversion 61

mapping of parameters for character conversion 29

mapping of parameters for conversion information

service 178

mapping of parameters for normalization 79

C interface for case conversion 60

C interface for character conversion 29

C interface for conversion information service 178

C interface for normalization 78

C: Customized Subset, value for *technique-character* in
CONVERSION 20

call syntax for case conversion 60, 62

call syntax for character conversion 29, 33

call syntax for conversion information service 178, 183

call syntax for normalization 78, 80

calling the stub routine

for character conversion 5

case conversion

ALET 60

C interface, call syntax 60

C interface, mapping of parameters 61

CASE control statement 59

definition in glossary 489

general description 5

HLASM interface, call syntax 62

HLASM interface, mapping of parameters 63, 70

interfaces, description of parameters 65, 71

mapping of parameters, C interface 61

mapping of parameters, HLASM interface 63, 70

parameters in area CUN4BAPR 71

parameters in area CUNBAPRM 65

restrictions of the calling environment 60

return and reason codes 457

return codes, classification 457

stub routine 59

case conversion handle 60

CUN4BAPR_Conv_Type 72

CUNBAPRM_Conv_Type 66

- CCSID
 - CCSIDs below X'DFFF' 25
 - converting strings of text characters 25
 - definition in glossary 489
 - indirect conversion 5
 - intermediate CCSID 5
 - intermediate CCSID, definition in glossary 491
 - MBCS conversions 5
 - range from X'E000' to X'FFFF' 25
- CDRA
 - definition in glossary 489
- character
 - definition in glossary 489
- character conversion
 - activating the conversion environment 25
 - ALET 28
 - big endian encoding 25
 - C interface, call syntax 29
 - C interface, mapping of parameters 29
 - calling the stub routine 5
 - conversion between CCSIDs 4
 - conversion handle 28
 - conversion types 4
 - definition in glossary 489
 - general description 4, 25
 - HLASM interface, call syntax 33
 - HLASM interface, mapping of parameters 34, 44
 - indirect conversion 26
 - interfaces, description of parameters 36, 47
 - items to be provided by caller 26
 - mapping of parameters, C interface 29
 - mapping of parameters, HLASM interface 34, 44
 - parameters in area CUN4BCPR 47
 - parameters in area CUNBCPRM 36
 - restrictions of the calling environment 29
 - return and reason codes 457
 - return codes, classification 457
 - UCCE, CUN4BCPR_Conv_Handle 48
 - UCCE, CUNBCPRM_Conv_Handle 37
 - UFT-8 25
- character conversion handle
 - case conversion 28
- character data representation architecture
 - see 'CCSID' definition in glossary 489
 - see 'CDRA' definition in glossary 489
- character set
 - definition in glossary 489
- code
 - definition in glossary 489
- code page
 - definition in glossary 489
 - source code page, see *FROM-CCSID* 28
 - target code page, see *TO-CCSID* 28
- code page conversion
 - definition in glossary 489
- code page global identifier
 - see 'CPGID' definition in glossary 490
- code point
 - definition in glossary 490
- code set
 - definition in glossary 490
- coded character
 - definition in glossary 490
- coded character set
 - definition in glossary 490
- codes
 - list of z/OS support for Unicode codes 457
- collation
 - general description 5
 - HLASM interface, mapping of parameters 110
 - interfaces, description of parameters 113, 133
 - mapping of parameters, HLASM interface 110
 - parameters in area CUN4BOPR 133
 - parameters in area CUNBOPRM 113
- collation level
 - definition in glossary 490
- collation rules
 - definition in glossary 490
- commands
 - syntax definitions xii
- composite conversion
 - definition in glossary 490
- composition 77
- control character
 - definition in glossary 490
- control function
 - definition in glossary 490
- control parameters
 - restriction while calling the case conversion services 60
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 178
- conversion environment
 - definition in glossary 490
- conversion handle
 - case conversion 28, 60
 - case conversion, CUN4BAPR_Conv_Type 72
 - case conversion, CUNBAPRM_Conv_Type 66
 - critical case when invalid 23
- conversion image 2
 - amount of storage needed 21
 - creating 9
 - definition in glossary 490
 - general description 9
 - layout 9
 - loading into storage 9
- Conversion information
 - general description 6
- conversion information service
 - C interface, call syntax 178
 - C interface, mapping of parameters 178
 - CCSID information 177
 - HLASM interface, call syntax 183
 - HLASM interface, mapping of parameters 188, 202
 - interfaces, description of parameters 193, 207
 - mapping of parameters, C interface 178
 - mapping of parameters, HLASM interface 188, 202
 - parameters in area CUN4BIPR 207
 - parameters in area CUNBIPRM 193
 - restrictions of the calling environment 177

- conversion information service (*continued*)
 - stub routine 177
- conversion of data between specified CCSIDs 25
- conversion service 177
- conversion services
 - definition in glossary 490
 - general description 3
- conversion tables
 - creating 452
 - creating a character map 452
 - defining 449
 - input to image generator 10
 - modifying job CUNJIUTL 454
 - provided by Unicode Consortium 5
 - user-defined 455
- conversion to upper or lower case 59
- conversion type
 - conversion types of CCSID conversions 4
 - in character conversion,
 - CUN4BAPR_Conv_Type 72
 - in character conversion,
 - CUNBAPRM_Conv_Type 66
- CPGID
 - definition in glossary 490
- creating a conversion table 452
- cross memory mode
 - restriction while calling the case conversion services 60
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 178
- CUN_RC_ENV_ERR, return code 457
- CUN_RC_OK, return code 457
- CUN_RC_SYS_ERR, return code 457
- CUN_RC_USER_WARN, return code 457
- CUN_RC_WARN, return code 457
- CUN4BAPR parameter area for case conversion 71
- CUN4BCPR parameter area for character conversion 47
- CUN4BIPR parameter area for conversion information service 207
- CUN4BIPR_CCSID1_ES 207, 211
- CUN4BNPR parameter area for normalization 87
- CUN4BOPR parameter area for collation 133
- CUNAIKBG macro 449
- CUNBA_DDA_req, constant 59
- CUNBAPRM parameter area for case conversion 65
- CUNBAPRM_Src_Buf_ALET 65
- CUNBAPRM_Src_Buf_Len 65
- CUNBAPRM_Targ_Buf_ALET 66
- CUNBAPRM_Targ_Buf_Len 66
- CUNBAPRM_Targ_Buf_Ptr 66
- CUNBCPRM parameter area for character conversion 36
- CUNBCPRM_Designator 42
- CUNBIPRM parameter area for conversion information service 193
- CUNBIPRM_CCSID1_ES 193, 197
- CUNBN_DDA_req, constant 78
- CUNBNPRM parameter area for normalization 83

- CUNBOPRM parameter area for collation 113
- CUNJIUTL
 - user-defined conversions 454
- CUNMIKBS macro 451

D

- DBCS
 - definition in glossary 490
 - indirect conversion 26
- DDA, see dynamic data area
 - constant CUNBAPRM_DDA_Req 59
 - constant CUNBNPRM_DDA_Req 78
- decomposition 77
- defining CCSIDs and conversion tables 449
- designator sequence 28
 - definition in glossary 490
- direct conversion 19
 - definition in glossary 490
- disability 483
- dispatchable unit mode
 - restriction while calling the case conversion services 60
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 177
- DISPLAY UNI command
 - syntax definitions xii
- double-byte (coded) character set
 - see 'DBCS' definition in glossary 490
- dynamic data area
 - case conversion 59, 78

E

- E: Enforced Subset, value for *technique-character* in CONVERSION 20
- EBCDIC
 - definition in glossary 490
- empty conversion environment
 - definition in glossary 490
- empty image
 - definition in glossary 490
- encoding scheme
 - definition in glossary 490
- encoding scheme identifier
 - definition in glossary 490
- endian
 - definition in glossary 490
- endian form
 - in character conversion, big endian 25
 - in UCS-2, big or little endian 4
- enforced subset
 - definition in glossary 490
- EUC
 - definition in glossary 490
- external interfaces for case conversion 60, 62
- external interfaces for character conversion 29, 33
- external interfaces for conversion information service 178, 183

external interfaces for normalization 78, 80

F

FROM-CCSID

- definition 28
- definition in glossary 491
- in character conversion 28
- to be specified in user-defined conversion tables 455

functions, overview 3

G

GB18030

- definition in glossary 491

graphic character

- definition in glossary 491

graphic character set

- definition in glossary 491

H

HASN mode

- restriction while calling the case conversion services 60
- restriction while calling the character conversion services 29
- restriction while calling the conversion information service 178

high-surrogate

- definition in glossary 491

HLASM interface

- mapping of parameters for case conversion 63, 70
- mapping of parameters for character conversion 34, 44
- mapping of parameters for collation 110
- mapping of parameters for conversion information service 188, 202
- mapping of parameters for normalization 82, 86

HLASM interface for case conversion 62

HLASM interface for character conversion 33

HLASM interface for conversion information

- service 183

HLASM interface for normalization 80

I

IBM Extended Binary Coded Decimal Interchange Code

- see 'EBCDIC' definition in glossary 490

IDF

- see *interface definition file* 491

IEASYSxx

- editing 7

image generator

- definition in glossary 491
- input 9
- return and reason codes 467

indirect conversion 5, 20

indirect conversion (*continued*)

- between mixed code pages and anything else than SBCS 26
- between TBCS and anything else than DBCS 26
- between UTF-8 and anything else than DBCS 26
- definition in glossary 491

infrastructure

- definition in glossary 491
- general description 2

interface definition file

- constant CUNBN_DDA_req 78
- CUNBNIDF 78

- definition in glossary 491

intermediate CCSID

- definition in glossary 491

intermediate CCSID 1200 5

interrupt status

- restriction while calling the case conversion services 60
- restriction while calling the character conversion services 29
- restriction while calling the conversion information service 178

invalid conversion handle 23

ISO/IEC 10646

- general description 1

J

Japanese messages 6

K

keyboard 483

knowledge base

- input to image generator 10
- module CUNAIKBG 449
- module CUNMIKBS 449
- user-defined conversion tables 455

L

L: Language Environment-Behavior, value for *technique-character* in CONVERSION 20

little endian 4

- definition in glossary 491

locks

- restriction while calling the character conversion services 29

Locks

- restriction while calling the case conversion services 60
- restriction while calling the conversion information service 178

low-surrogate

- definition in glossary 491

lowercase

- definition in glossary 491

M

M: Modified Language Environment-Behavior, value for *technique-character* in CONVERSION 20

macro

CUNAIKBG 449

CUNMIKBS 451

malformed character

definition in glossary 491

mapping of parameters

for case conversion 61, 63, 70

for character conversion 29, 34, 44

for collation 110

for conversion information service 178, 188, 202

for normalization 79, 82, 86

mapping of parameters in C interface, for case conversion 61

mapping of parameters in C interface, for character conversion 29

mapping of parameters in C interface, for conversion information service 178

mapping of parameters in C interface, for normalization 79

mapping of parameters in HLASM interface, for case conversion 63, 70

mapping of parameters in HLASM interface, for character conversion 34, 44

mapping of parameters in HLASM interface, for collation 110

mapping of parameters in HLASM interface, for conversion information service 188, 202

mapping of parameters in HLASM interface, for normalization 82, 86

MBCS

definition in glossary 491

internal handling of MBCS conversions

detailed description 219

general description 5

illustration of MBCS decomposition 218

mixed code page

definition in glossary 491

mixed code pages 4

mode

amode 29, 60, 178

AR mode 29, 60, 178

ASC mode 29, 60, 178

cross memory mode 29, 60, 178

dispatchable unit mode 29, 60, 177

HASN mode 29, 60, 178

PASN mode 29, 60, 178

SASN mode 29, 60, 178

multi-byte character set

see 'MBCS' definition in glossary 491

N

naming convention

user-defined conversion tables 455

NLS

Japanese messages 6

normalization

C interface, call syntax 78

C interface, mapping of parameters 79

definition in glossary 491

general description 5

HLASM interface, call syntax 80

HLASM interface, mapping of parameters 82, 86

interfaces, description of parameters 83, 87

mapping of parameters, C interface 79

mapping of parameters, HLASM interface 82, 86

parameters in area CUN4BNPR 87

parameters in area CUNBNPRM 83

stub routine 77

normalization form

definition in glossary 491

normalization form C

definition in glossary 491

normalization form D

definition in glossary 492

normalization form KC

definition in glossary 492

normalization form KD

definition in glossary 492

Notices 485

O

overview of functions 3

P

parameter

description of parameters, case conversion 65, 71

description of parameters, character conversion 36, 47

description of parameters, collation 113, 133

description of parameters, conversion information service 193, 207

description of parameters, normalization 83, 87

parameter area

CUN4BAPR for case conversion 71

CUN4BCPR for character conversion 47

CUN4BIPR for conversion information service 207

CUN4BNPR for normalization 87

CUN4BOPR for collation 133

CUNBAPRM for case conversion 65

CUNBCPRM for character conversion 36

CUNBIPRM for conversion information service 193

CUNBNPRM for normalization 83

CUNBOPRM for collation 113

PASN mode

restriction while calling the case conversion services 60

restriction while calling the character conversion services 29

restriction while calling the conversion information service 178

PC

definition in glossary 492

personal computer

see 'PC' definition in glossary 492

- primary address space
 - restriction while calling the case conversion services 60
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 178
- problem determination
 - invalid conversion handle 23
 - target buffer overflow 55
 - work buffer overflow 78
- problem state
 - restriction while calling the case conversion services 60
 - restriction while calling the conversion information service 177
- programing interfaces for case conversion 60, 62
- programing interfaces for character conversion 29, 33
- programing interfaces for conversion information service 178, 183
- programing interfaces for normalization 78, 80
- PSW key
 - restriction while calling the case conversion services 60
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 177

Q

- QBCS
 - definition in glossary 492
- quadruple-byte character set
 - see 'QBCS' definition in glossary 492

R

- R: Roundtrip, value for *technique-character* in CONVERSION 20
- range of CCSIDs
 - CCSIDs below X'DFFF' (standard CCSIDs) 25
 - range from X'E000' to X'FFFF' (user-defined CCSIDs) 25
- reason code
 - reason code CUN_RS_TRG_EXH (target buffer exhausted) 55
- reason codes
 - conversion services 457
- recommendations
 - conversion handle 23
 - for the calling environment (case conversion) 60
 - for the calling environment (character conversion) 29
 - for the calling environment (conversion information service) 177
 - target buffer size 55
 - work buffer size 78
- recovery environment
 - restriction while calling the case conversion services 60

- recovery environment (*continued*)
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 178
- restrictions
 - case conversion 60
 - character conversion 29
 - conversion information service 177
- return codes
 - conversion services 457
 - image generator 467
- Round trip
 - definition in glossary 492

S

- sample program
 - case conversion 76
 - character conversion 57
 - collation 152
 - conversion information service 215
 - normalization 89
 - stringprep conversion 176
- SASN mode
 - restriction while calling the case conversion services 60
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 178
- SBCS
 - definition in glossary 492
 - indirect conversion 26
- script
 - definition in glossary 492
- SET UNI command
 - syntax definitions xii
- shortcut keys 483
- simple code page
 - definition in glossary 492
- simple code pages 4
- simple conversion
 - definition in glossary 492
- single-byte character set
 - see 'SBCS' definition in glossary 492
- sort key
 - definition in glossary 492
- source code page, see *FROM-CCSID* 28
- SRB or task
 - restriction while calling the case conversion services 60
 - restriction while calling the character conversion services 29
 - restriction while calling the conversion information service 177
- standards
 - ISO/IEC 10646 1
 - Unicode Standard 1
- storage
 - needed for a conversion image 21

- Stringprep
 - general description 6
- stub routine
 - CUN4LCNV 25
 - CUNLASE 59
 - CUNLCNV 25
 - CUNLINFO 177
 - CUNLNORM 77
- sub code page
 - definition in glossary 492
- supervisor state
 - restriction while calling the case conversion
 - services 60
 - restriction while calling the conversion information
 - service 177
- surrogate 4
- surrogate pair 492

T

- target buffer
 - calculating the size 55
 - reason code CUN_RS_TRG_EXH (target buffer exhausted) 55
- target code page, see *TO-CCSID* 28
- task or SRB
 - restriction while calling the case conversion
 - services 60
 - restriction while calling the character conversion
 - services 29
 - restriction while calling the conversion information
 - service 177
- TBCS
 - definition in glossary 492
 - indirect conversion 26
- TO-CCSID
 - definition 28
 - definition in glossary 492
 - in character conversion 28
 - user-defined conversion tables 455
- triple-byte character set
 - see 'TBCS' definition in glossary 492
- two-stage conversion, see indirect conversion 26

U

- UCAE
 - definition in glossary 492
- UCCB
 - definition in glossary 492
- UCCE
 - character conversion 25
 - character conversion,
 - CUN4BCPR_Conv_Handle 48
 - character conversion,
 - CUNBCPRM_Conv_Handle 37
 - definition in glossary 492
 - handle to UCCE instead of CCSIDs 25
- UCS
 - definition in glossary 492

- UCS transformation format
 - see 'UTF-16' definition in glossary 493
 - see 'UTF-32' definition in glossary 493
 - see 'UTF-8' definition in glossary 493
- UCS-2
 - big endian, little endian 4
 - definition in glossary 492
- Unicode character conversion control entry
 - see 'UCCE' definition in glossary 492
- Unicode Consortium
 - conversion tables provided by 5
- Unicode environment
 - general description 2
- Unicode standard
 - range of CCSIDs from X'E000' to X'FFFF' 25
- Unicode Standard
 - definition in glossary 492
 - general description 1
- Unicode transformation format
 - see 'UTF-16' definition in glossary 493
 - see 'UTF-32' definition in glossary 493
 - see 'UTF-8' definition in glossary 493
- Unicode® Consortium
 - Web site xii
- universal character set
 - see 'UCS' definition in glossary 492
- uppercase
 - definition in glossary 492
- user-defined CCSID
 - valid range X'E000' to X'FFFF' 25
- user-defined conversion tables 455
- UTF-16
 - definition in glossary 493
 - in ISO/IEC 10646 1
 - in Unicode Standard 1
- UTF-32
 - definition in glossary 493
- UTF-8
 - definition in glossary 493
 - indirect conversion 26

W

- weight
 - definition in glossary 493
- work buffer
 - calculating the size 78

Z

- z/OS support for Unicode
 - case conversion, general description 5
 - character conversion, general description 4
 - conversion services, general description 3
 - conversion to upper or lower case 5
 - conversion types 4
 - general description 1
 - general description of component: conversion
 - services 2
 - general description of component: infrastructure 2
 - infrastructure, general description 2

z/OS support for Unicode (*continued*)

- MBCS conversions, general description 217
- messages, Japanese 6
- prerequisites 7
- return and reason codes 457
- return and reason codes from conversion services 457
- return codes from image generator 467
- UCS-2, big endian 4
- Unicode environment, general description 2

Readers' Comments — We'd Like to Hear from You

z/OS

Support for Unicode:

Using Unicode Services

Publication No. SA22-7649-10

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: mhvrfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



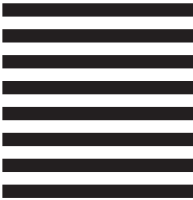
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY
12601-5400



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5694-A01

Printed in USA

SA22-7649-10

