

z/OS
3.2

*Unicode Services User's Guide and
Reference*



Note

Before using this information and the product it supports, read the information in [“Notices” on page 613.](#)

This edition applies to IBM® z/OS® 3.2 (5655-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-30

© **Copyright International Business Machines Corporation 2001, 2025.**

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

Contents

| | |
|--|-------------|
| Figures..... | ix |
| Tables..... | xi |
| About this information..... | xv |
| Who should use this information..... | xv |
| How this information is organized..... | xv |
| Overview of contents..... | xv |
| Syntax diagrams..... | xvi |
| z/OS information..... | xvi |
| How to provide feedback to IBM..... | xvii |
| Summary of changes..... | xix |
| Summary of changes for z/OS 3.2..... | xix |
| Summary of changes for z/OS 3.1..... | xix |
| Part 1. Introduction to the Unicode Standard and z/OS Unicode Services..... | 1 |
| Chapter 1. Introduction to the Unicode Standard..... | 3 |
| What is the Unicode Standard?..... | 3 |
| The Unicode standard..... | 3 |
| How the Unicode Standard relates to prior standards such as ASCII and EBCDIC..... | 4 |
| Evolving standards based on limited platforms..... | 4 |
| Historical simplicity creates modern complexity..... | 4 |
| Character sets for many characters..... | 4 |
| Stateful encodings..... | 5 |
| Why the Unicode Standard?..... | 5 |
| What is z/OS Unicode Services?..... | 5 |
| z/OS support for the Unicode Standard application programming interfaces..... | 6 |
| Character conversion..... | 6 |
| Case conversion..... | 7 |
| Normalization..... | 7 |
| Collation..... | 7 |
| Stringprep..... | 7 |
| Bidirectional transformation..... | 7 |
| Conversion information service..... | 8 |
| Dynamic locale service..... | 8 |
| Part 2. Application programmer information..... | 9 |
| Chapter 2. About the application programming interfaces | 11 |
| z/OS Unicode environment..... | 11 |
| General concepts when using the z/OS Unicode Services programming interfaces..... | 11 |
| Conversion handle use..... | 13 |
| Sample code..... | 14 |
| Characteristics for the caller | 14 |
| Linkage conventions..... | 14 |
| Bidi function..... | 14 |

| | |
|--|-----|
| Related services..... | 14 |
| Chapter 3. Character conversion..... | 15 |
| Calling the character conversion services..... | 15 |
| Calling the bidi conversion services..... | 17 |
| Restrictions for the calling environment..... | 19 |
| Using the C interface..... | 19 |
| Mapping of parameters in C..... | 19 |
| 31-bit mapping..... | 19 |
| 64-bit mapping..... | 22 |
| Using the HLASM interface..... | 24 |
| Mapping of parameters for AMODE (31)..... | 25 |
| Description of parameters in area CUNBCPRM..... | 28 |
| Mapping of parameters for AMODE (64)..... | 35 |
| Description of parameters in area CUN4BCPR..... | 38 |
| Mapping of the extended bidi parameter area..... | 45 |
| AMODE(31)..... | 46 |
| Description of parameters in area CUNBDPRM..... | 48 |
| AMODE(64)..... | 61 |
| Description of parameters in area CUN4BDPR..... | 64 |
| Handling a target buffer overflow | 76 |
| Sample programs..... | 78 |
| Chapter 4. Case conversion..... | 79 |
| Calling the case conversion services..... | 80 |
| Restrictions for the calling environment..... | 81 |
| Using the C interface..... | 81 |
| Mapping of parameters in C..... | 81 |
| 31-bit mapping..... | 81 |
| 64-bit mapping..... | 82 |
| Using the HLASM interface..... | 83 |
| Mapping of parameters for AMODE (31)..... | 84 |
| Description of parameters in area CUNBAPRM..... | 85 |
| Mapping of parameters for AMODE (64)..... | 89 |
| Description of parameters in area CUN4BAPR..... | 91 |
| Sample programs..... | 95 |
| Chapter 5. Normalization..... | 97 |
| Calling the normalization service..... | 97 |
| Handling a work buffer overflow | 98 |
| Restrictions for the calling environment..... | 98 |
| Using the C interface..... | 98 |
| Mapping of parameters in C..... | 99 |
| 31-bit mapping..... | 99 |
| 64-bit mapping..... | 99 |
| Using the HLASM interface..... | 100 |
| Mapping of parameters for AMODE (31)..... | 101 |
| Description of parameters in area CUNBNPRM..... | 102 |
| Mapping of parameters for AMODE (64)..... | 105 |
| Description of parameters in area CUN4BNPR..... | 106 |
| Sample programs..... | 108 |
| Chapter 6. Collation..... | 111 |
| Calling the collation service..... | 112 |
| Restrictions for the calling environment..... | 116 |
| Using the C interface..... | 116 |
| Mapping of parameters in C..... | 120 |
| 31-bit mapping..... | 120 |

| | |
|--|-----|
| 64-bit mapping..... | 122 |
| Mapping of constants in C..... | 123 |
| Using the HLASM interface..... | 125 |
| Mapping of parameters for AMODE (31)..... | 127 |
| Description of parameters in area CUNBOPRM..... | 131 |
| Mapping of constants for AMODE (31)..... | 144 |
| Mapping of parameters for AMODE (64)..... | 146 |
| Description of parameters in area CUN4BOPR..... | 149 |
| Mapping of constants for AMODE (64)..... | 162 |
| Sort key vector format..... | 163 |
| Work buffer length considerations..... | 164 |
| Target buffer length considerations..... | 165 |
| Sample programs..... | 166 |
| Chapter 7. Bidi transformation..... | 169 |
| Calling bidi transformation service..... | 169 |
| Using the C interface..... | 169 |
| Mapping of parameters in C..... | 170 |
| 31-bit mapping..... | 170 |
| 64-bit mapping..... | 171 |
| Using the HLASM interface..... | 171 |
| Mapping of parameters for AMODE (31)..... | 172 |
| Description of parameters in area CUNBBPRM..... | 173 |
| Mapping of parameters for AMODE (64)..... | 174 |
| Description of parameters in area CUN4BBPR..... | 175 |
| Character conversion service and the new B technique..... | 176 |
| Chapter 8. Stringprep conversion..... | 179 |
| Calling the stringprep services..... | 179 |
| Using the C interface..... | 179 |
| Mapping of parameters in C..... | 180 |
| 31-bit mapping..... | 180 |
| 64-bit mapping..... | 181 |
| Using the HLASM interface..... | 181 |
| Mapping of parameters for AMODE (31)..... | 182 |
| Description of parameters in area CUNBPPRM..... | 184 |
| Mapping of parameters for AMODE (64)..... | 186 |
| Description of parameters in area CUN4BPPR..... | 187 |
| Sample programs..... | 189 |
| Chapter 9. Conversion information service..... | 191 |
| Calling the conversion information service..... | 191 |
| Restrictions for the calling environment..... | 192 |
| Using the C interface..... | 192 |
| Mapping of parameters in C..... | 192 |
| 31-bit mapping..... | 192 |
| 64-bit mapping..... | 195 |
| Using the HLASM interface..... | 198 |
| Mapping of parameters for AMODE (31)..... | 202 |
| Description of parameters in area CUNBIPRM..... | 207 |
| Mapping of parameters for AMODE (64)..... | 216 |
| Description of parameters in area CUN4BIPR..... | 221 |
| Sample programs..... | 229 |
| Chapter 10. Dynamic locale service..... | 231 |
| Adding and removing locales in the z/OS Unicode environment..... | 231 |
| Mapping of parameters in C..... | 231 |
| 31-bit mapping..... | 231 |

| | |
|--|------------|
| 64-bit mapping..... | 232 |
| Mapping of parameters for AMODE (31)..... | 233 |
| Description of parameters in area CUNBLPRM..... | 234 |
| Mapping of parameters for AMODE (64)..... | 236 |
| Description of parameters in area CUN4BLPR..... | 238 |
| Part 3. System programmer information..... | 241 |
| Chapter 11. z/OS Unicode environment..... | 243 |
| Key concepts behind the z/OS Unicode environment..... | 243 |
| Life cycle..... | 243 |
| Dynamic loading..... | 243 |
| CUNUNIXx parmlib statements..... | 243 |
| The knowledge base..... | 244 |
| The SETUNI command..... | 244 |
| Equivalent commands..... | 245 |
| The DISPLAY UNI command..... | 245 |
| How conversions are deleted from the z/OS Unicode environment..... | 245 |
| Storage requirements..... | 245 |
| Page-fixed (REALSTORAGE)..... | 246 |
| Conversion images..... | 246 |
| The DB2 conversion image..... | 247 |
| Chapter 12. Diagnostic tools for z/OS Unicode environment errors..... | 249 |
| Diagnosing Unicode environment errors..... | 249 |
| API return codes..... | 249 |
| Console messages..... | 249 |
| The DISPLAY UNI command..... | 249 |
| The z/OS Unicode environment mapping utility (CUNMIMAP)..... | 249 |
| Dumping the z/OS Unicode dataspace..... | 250 |
| Recovering from z/OS Unicode environment errors..... | 250 |
| Invalid conversion handles..... | 251 |
| Chapter 13. Manually setting up z/OS Unicode Services..... | 253 |
| Prerequisites..... | 253 |
| Configuring the z/OS Unicode environment..... | 253 |
| Updating parmlib members..... | 253 |
| MVS Message Service..... | 253 |
| Creating the z/OS Unicode Services environment..... | 254 |
| Creating a conversion image..... | 254 |
| Chapter 14. Creating user-defined conversion tables..... | 269 |
| Format of tables..... | 269 |
| Table naming convention..... | 269 |
| Creating a user-defined conversion table between two existing CCSIDs..... | 270 |
| Example of building a character map based from an existing conversion table..... | 271 |
| Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID..... | 276 |
| Step 1: Update the z/OS Unicode Services knowledge base..... | 277 |
| Steps two through seven..... | 279 |
| Reference information about the CUNAIKBG macro..... | 279 |
| Chapter 15. Unicode batch tools..... | 281 |
| CUNMTUNI: Unicode Services batch client..... | 281 |
| Example..... | 281 |
| User interface..... | 281 |
| CUNMCSMX/CUNMRCSX: sending MIME data via SMTP..... | 283 |

| | |
|---|------------|
| Send mail from the batch..... | 283 |
| CUNMCSMH/CUNMRCSH: sending HTML data via SMTP..... | 286 |
| Send mail from the batch..... | 287 |
| CUNMCBLI: the COBOL API utility for sending MIME data via SMTP..... | 289 |
| Example..... | 297 |
| User Interface..... | 297 |
| CUNMCSMM/CUNMRCSM: sending MIME data via SMTP..... | 298 |
| Send mail from the batch..... | 298 |
| Appendix A. Description of CCSIDs..... | 309 |
| Unicode CCSIDs..... | 309 |
| Encoding Scheme..... | 309 |
| Appendix B. Conversion support for multi-byte encodings (MBCS)..... | 329 |
| Internal handling of MBCS conversions..... | 329 |
| MBCS CCSID decomposition..... | 329 |
| When a shift character is in the data stream..... | 332 |
| MBCS CCSIDs compatible with iconv..... | 333 |
| C-variant MBCS CCSIDs compatible with iconv()..... | 333 |
| Appendix C. Conversion tables supplied with z/OS Unicode Services..... | 335 |
| Direct conversions supported between non-Unicode CCSIDs | 335 |
| Direct conversions supported to and from Unicode..... | 507 |
| Appendix D. Validation, case, normalization, collation, & stringprep resources.... | 523 |
| Validation tables..... | 523 |
| Case conversion tables..... | 524 |
| Normalization tables..... | 528 |
| Collation tables..... | 531 |
| Stringprep tables..... | 535 |
| Appendix E. Locales for collation and case support..... | 537 |
| Locales supported for collation..... | 537 |
| Locales supported for case service..... | 554 |
| Appendix F. Locales for dynamic locale service..... | 557 |
| Adding and removing locales to the z/OS Unicode Services environment..... | 569 |
| Euro and pre-euro support..... | 570 |
| Language Environment C/C++ Runtime Library compatible locale support..... | 570 |
| Appendix G. System control offsets..... | 595 |
| Examples for 31-bit callers..... | 595 |
| List of offsets for 31-bit services..... | 595 |
| Examples for 64-bit callers..... | 595 |
| List of offsets for 64-bit services..... | 595 |
| Appendix H. Unicode return and reason codes..... | 597 |
| Return code meanings..... | 597 |
| Image generator for z/OS support for Unicode – return codes..... | 608 |
| Appendix I. Accessibility..... | 611 |
| Notices..... | 613 |
| Terms and conditions for product documentation..... | 614 |
| IBM Online Privacy Statement..... | 615 |
| Policy for unsupported hardware..... | 615 |

| | |
|---|------------|
| Minimum supported hardware..... | 615 |
| Trademarks..... | 616 |
| Glossary of terms and abbreviations..... | 617 |
| Index..... | 625 |

Figures

| | |
|---|-----|
| 1. Conversion of MBCS data to Unicode characters..... | 332 |
|---|-----|

Tables

| | |
|--|-----|
| 1. Restrictions while calling the character conversion services..... | 19 |
| 2. Mapping of parameters in HLASM for character conversion AMODE (31) | 25 |
| 3. Mapping of parameters in HLASM for character conversion AMODE (64)..... | 35 |
| 4. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(31)..... | 46 |
| 5. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(64)..... | 61 |
| 6. Minimum and maximum character widths of the different encoding schemes..... | 77 |
| 7. Word delimiter characters..... | 79 |
| 8. Restrictions while calling the case conversion services..... | 81 |
| 9. Mapping of parameters in HLASM for case conversion AMODE (31)..... | 84 |
| 10. Mapping of parameters in HLASM for case conversion AMODE (64)..... | 89 |
| 11. Unicode version table..... | 97 |
| 12. Restrictions while calling the normalization service..... | 98 |
| 13. Mapping of parameters in HLASM for normalization AMODE (31)..... | 101 |
| 14. Mapping of parameters in HLASM for normalization AMODE (64)..... | 105 |
| 15. Restrictions for the calling environment..... | 116 |
| 16. Mapping of parameters in HLASM for collation AMODE (31)..... | 128 |
| 17. Collation mask sub fields descriptions..... | 134 |
| 18. Equivalencies between short path and long path locale settings..... | 138 |
| 19. Collation keywords descriptions..... | 138 |
| 20. Valid values for collation keywords..... | 140 |
| 21. Collation rule symbols..... | 141 |
| 22. Collation syntax rules..... | 142 |

| | |
|---|-----|
| 23. Mapping of parameters in HLASM for collation AMODE (64)..... | 146 |
| 24. Collation mask sub fields descriptions..... | 153 |
| 25. Equivalencies between short path and long path local settings..... | 156 |
| 26. Collation keywords descriptions..... | 156 |
| 27. Valid values for collation keywords..... | 158 |
| 28. Collation rule symbols..... | 159 |
| 29. Collation syntax rules..... | 160 |
| 30. Collation level weight length..... | 164 |
| 31. Size of the work buffers for UTF-16BE Code Points..... | 164 |
| 32. Recommended target buffer lengths for collation..... | 165 |
| 33. Size of the target buffers for UTF-16BE Code Points | 165 |
| 34. Target Buffer Formula..... | 166 |
| 35. The AMODE and API (C/C++ or HLASM) in combination with long or short path settings..... | 166 |
| 36. Mapping of parameters in HLASM for bidi AMODE (31)..... | 172 |
| 37. Mapping of parameters in HLASM for bidi AMODE (64)..... | 174 |
| 38. Mapping of parameters in HLASM for stringprep AMODE (31)..... | 182 |
| 39. Mapping of parameters in HLASM for stringprep AMODE (64)..... | 186 |
| 40. Restrictions while calling the conversion information service services..... | 192 |
| 41. Mapping of parameters in HLASM for conversion information service AMODE (31)..... | 202 |
| 42. Mapping of parameters in HLASM for conversion information service AMODE (64)..... | 216 |
| 43. Mapping of parameters in dynamic locale service AMODE (31)..... | 233 |
| 44. Mapping of parameters in dynamic locale service AMODE (64)..... | 236 |
| 45. Main storage needed for conversions of type SBCS and DBCS..... | 265 |
| 46. Main storage needed for conversions of type MBCS..... | 266 |
| 47. CUNMRCSX usable formats..... | 284 |

| | |
|--|-----|
| 48. Data type descriptions..... | 286 |
| 49. Return codes for CUNMRCSX..... | 286 |
| 50. Usable formats for CUNMRCSH..... | 287 |
| 51. Data type descriptions..... | 289 |
| 52. Return codes for CUNMRCSH..... | 289 |
| 53. Copybook description..... | 291 |
| 54. Usable formats for CUNMRCSM..... | 298 |
| 55. Data type descriptions..... | 303 |
| 56. Comments in PARMDD..... | 304 |
| 57. Return codes for CUNMRCSM..... | 306 |
| 58. Encoding schemes..... | 309 |
| 59. CCSIDs supported by z/OS Unicode..... | 310 |
| 60. CCSID conversions types of z/OS support for the Unicode Standard | 335 |
| 61. Non-Unicode Conversions Available..... | 335 |
| 62. Direct Conversions Supported to and from Unicode CCSID 01200..... | 507 |
| 63. Character conversion service supporting validation..... | 523 |
| 64. Case conversion service based on the Unicode Standard 3.0.1..... | 524 |
| 65. Case conversion service based on the Unicode Standard 3.2.0..... | 525 |
| 66. Case conversion service based on the Unicode Standard 4.0.1..... | 525 |
| 67. Case conversion service based on the Unicode Standard 4.1.0..... | 525 |
| 68. Case conversion service based on the Unicode Standard 5.0.0..... | 526 |
| 69. Case conversion service based on the Unicode Standard 6.0.0..... | 526 |
| 70. Case conversion service based on the Unicode Standard 9.0.0..... | 527 |
| 71. Case conversion service based on the Unicode Standard 13.0.0..... | 527 |
| 72. Normalization service based on the Unicode Standard 3.0.1..... | 528 |

| | |
|--|-----|
| 73. Normalization service based on the Unicode Standard 3.2.0. | 528 |
| 74. Normalization service based on the Unicode Standard 4.0.1. | 528 |
| 75. Normalization service based on the Unicode Standard 4.1.0. | 529 |
| 76. Normalization service based on the Unicode Standard 6.0.0. | 530 |
| 77. Normalization service based on the Unicode Standard 9.0.0. | 530 |
| 78. Normalization service based on the Unicode Standard 13.0.0. | 531 |
| 79. Collation service based on the Unicode Standard 3.0.1..... | 531 |
| 80. Collation service based on the Unicode Standard 4.0.0..... | 532 |
| 81. Collation service based on the Unicode Standard 4.1.0..... | 533 |
| 82. Collation service based on the Unicode Standard 6.0.0..... | 533 |
| 83. Collation service based on the Unicode Standard 9.0.0..... | 534 |
| 84. Collation service based on the Unicode Standard 13.0.0..... | 535 |
| 85. Profiles provided for stringprep service..... | 536 |
| 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit)... | 537 |
| 87. Case service and locale valid names..... | 554 |
| 88. z/OS Unicode Services locale standard source repository..... | 557 |
| 89. Offsets for 31-bit callers..... | 595 |
| 90. Offsets for 64-bit callers..... | 595 |
| 91. Classification of return codes..... | 597 |
| 92. Return and reason codes from z/OS Unicode Services..... | 597 |

About this information

This information provides guidance for using z/OS support for the Unicode Standard.

Who should use this information

This information is intended for application programmers, system programmers, and system administrators who want to know how to set up and use the z/OS Unicode Services environment.

How this information is organized

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

Overview of contents

This document contains the following information:

- [Part 1, “Introduction to the Unicode Standard and z/OS Unicode Services,” on page 1](#)
 - Chapter 1, “Introduction to the Unicode Standard,” on page 3 is an overview of what the Unicode Standard is and what Unicode support on the z/OS platform is.
- [Part 2, “Application programmer information,” on page 9](#)
 - [Chapter 2, “About the application programming interfaces,” on page 11](#) describes the programming interfaces provided by z/OS Unicode Services.
 - [Chapter 3, “Character conversion,” on page 15](#) gives instructions on how to use the character conversion services.
 - [Chapter 4, “Case conversion,” on page 79](#) gives instructions on how to use the case conversion services.
 - [Chapter 5, “Normalization,” on page 97](#) gives instructions on how to use the normalization services.
 - [Chapter 6, “Collation,” on page 111](#) gives instructions on how to use the collation services.
 - [Chapter 7, “Bidi transformation,” on page 169](#) describes the programming required for the bidi transformation service.
 - [Chapter 8, “Stringprep conversion,” on page 179](#) describes the programming required for the stringprep conversion services.
 - [Chapter 9, “Conversion information service,” on page 191](#) describes the programming required for the conversion information service.
- [Part 3, “System programmer information,” on page 241](#)
 - [Chapter 11, “z/OS Unicode environment,” on page 243](#) describes the Unicode environment.
 - [Chapter 12, “Diagnostic tools for z/OS Unicode environment errors,” on page 249](#) describes how the system operator can recover from errors in the Unicode environment.
 - [Chapter 13, “Manually setting up z/OS Unicode Services,” on page 253](#) describes how to set up the system to use Unicode Services if you want to configure the system manually.
 - [Chapter 14, “Creating user-defined conversion tables,” on page 269](#) describes how to create user defined conversion tables and have Unicode Services Character Conversion Service use them.
 - [“Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID” on page 276](#) shows how you can define a user defined CCSID in the Unicode services knowledge base.
- [“Encoding Scheme” on page 309](#) describes the CCSIDs supported by the Unicode environment.

- [Appendix B, “Conversion support for multi-byte encodings \(MBCS\),” on page 329](#) describes how MBCS conversions are handled internally.
- [Appendix C, “Conversion tables supplied with z/OS Unicode Services,” on page 335](#) shows all tables IBM provides for conversions.
- [Appendix D, “Validation, case, normalization, collation, & stringprep resources,” on page 523](#) describes the conversion tables supplied by the Unicode environment.
- [Appendix E, “Locales for collation and case support,” on page 537](#) lists the locales supported in the data set SYS1.SCUNLOCL.
- [Appendix G, “System control offsets,” on page 595](#) describes the system control offsets that can be used as an alternative to linking or link-editing the service stub.
- [Appendix H, “Unicode return and reason codes,” on page 597](#) lists the Unicode Services return and reason codes.
- [Appendix I, “Accessibility,” on page 611](#) describe the major accessibility features in z/OS.
- [“Glossary of terms and abbreviations” on page 617](#) explains the terminology used in this document.

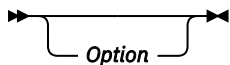
Syntax diagrams

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

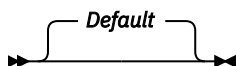
A command or keyword that you must enter (a required command) is displayed like this:

➡ Command ➡

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:



z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. For more information, see [How to send feedback to IBM](#).

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under [IBM z/OS Product Documentation Update Policy](http://www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy) (www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy).

Summary of changes for z/OS 3.2

The following content is new, changed, or no longer included in z/OS 3.2.

New

The following content is new.

September 2025 release

- None.

Changed

The following content is changed.

September 2025 release

- None.

Deleted

The following content is deleted.

September 2025 release

- None.

Summary of changes for z/OS 3.1

The following content is new, changed, or no longer included in z/OS 3.1.

New

The following content is new.

March 2024 refresh

For APAR OA65878, the following topics is updated: [“Encoding Scheme” on page 309](#) and [“MBCS CCSID decomposition” on page 329](#).

September 2023 release

- Support added for ISO-2022 JP standard. For more information, refer to [“31-bit mapping” on page 19](#) and [“64-bit mapping” on page 22](#).

Changed

The following content is changed.

September 2023 release

- None.

Deleted

The following content was deleted.

September 2023 release

- None.

Part 1. Introduction to the Unicode Standard and z/OS Unicode Services

Chapter 1. Introduction to the Unicode Standard

z/OS Unicode Services provides a set of functions that work with the Unicode Standard. This section describes the z/OS Unicode Services, what they contain, how to work with them and other related issues.

What is the Unicode Standard?

The Unicode Standard precisely defines a character set as well as a small number of encodings for it. It enables you to handle text in any language efficiently. It allows a single application to work for a global audience.

Before the Unicode Standard, the encoding systems that existed did not cover all the necessary numbers, characters, and symbols in use. Different encoding systems might assign the same number to different characters. If you used the wrong encoding system, your output might not have been what you expected to see.

The Unicode Standard provides a unique number for every character, regardless of platform, language, or program. Using the Unicode Standard, you can develop a software product that works with various platforms, languages, and countries. The Unicode Standard also allows data to be transported through many different systems. Modern systems provide internationalization solutions based on the Unicode Standard.

The original Unicode Standard repertoire covered all major languages commonly used in computing. The Unicode Standard continues to grow and to include more scripts.

The design of the Unicode Standard differs in several ways from traditional character sets and encoding schemes:

- Its repertoire enables users to include text efficiently in almost all languages within a single document.
- It can be encoded in a byte-based way with one or more bytes per character, but the default encoding scheme uses 16-bit units that allow much simpler processing for all common characters.
- Many characters, such as letters with accents and umlauts, can be combined from the base character and accent or umlaut modifiers. This combining reduces the number of different characters that need to be encoded separately. Pre-composed variants for characters that existed in common character sets at the time were included for compatibility.

Characters and their usage are well-defined and described. Traditional character sets typically provide only the name or a picture of a character and its number and byte encoding; the Unicode Standard has a comprehensive database of properties available. It also defines a number of processes and algorithms for dealing with many aspects of text processing to make it more interoperable.

The early inclusion of all characters of commonly used character sets makes the Unicode Standard a useful mechanism for converting between traditional character sets, and makes it feasible to process non-Unicode text by first converting the text into Unicode, processing the text, and then converting it back to the original encoding without loss of data.

The Unicode standard

The Unicode Standard has been adopted by such industry leaders as IBM Corporation, Google Inc, Apple, Inc., Microsoft Corporation, Oracle Corporation, and many other government and educational institutions.

The Unicode Standard is the foundation of modern computer standards and is the character infrastructure of the Internet and the World Wide Web. It is supported in many operating systems, all modern browsers, and many other products.

For more information on the Unicode Standard, see [The Unicode Consortium \(www.unicode.org\)](http://www.unicode.org).

How the Unicode Standard relates to prior standards such as ASCII and EBCDIC

The Unicode Standard has advantages over other standards. It can reduce the complexity of handling character data in globalized applications.

Evolving standards based on limited platforms

The representation of character data in modern computer systems can be fairly complicated, depending on the needs of your globalized application. One of the reasons for this complexity is that the methods for handling this data have evolved from early methods that served less complicated environments and hardware platforms.

In fact, many early decisions about how to encode characters on a system were guided by the functional requirements of specific devices, such as the early Telex (TTY) terminals and punch card technologies. For example, the Delete character (with an ASCII value of x'7F') was required in order to punch out all of the holes in a column of a punch card to signify that the column should be ignored. The storage capacities of these early computing systems placed additional limitations on system and application designers.

The character encoding schemes that have grown out of these early systems were built on this historical foundation:

- The ASCII (American Standard Code for Information Interchange) character set uses 7-bit units, with a trivial encoding designed for 7-bit bytes. It is the most important character set in use today, despite its limitation to very few characters, because its design is the foundation for most modern character sets. ASCII provides only 128 numeric values, and 33 of those are reserved for special functions.
- The EBCDIC (Extended Binary-Coded Decimal Interchange Code) character set and a number of associated character sets, designed by IBM for its mainframes, uses 8-bit bytes. It was developed at a similar time as ASCII, and shares the same set of base characters and has other similar properties. Unlike ASCII, the Latin letters are not combined in two blocks for upper- and lower-case. Instead, the letters are arranged so that their hexadecimal values have second digits of 1 through 9.

Historical simplicity creates modern complexity

The physical and functional limitations of the early character sets gave way to rapidly expanding hardware and functional capabilities. Character representation on computing systems became less dependent on hardware; instead, software designers used the existing encoding schemes to accommodate the needs of an increasingly global community of computer users.

Character sets for many characters

The most common encodings (character encoding schemes) use a single byte per character, and they are often called single-byte character sets (SBCS). They are all limited to 256 characters. Because of this, none of them can even cover all of the accented letters for the Western European languages. Consequently, many different such encodings were created over time to fulfill the needs of different user communities. The most widely used SBCS encoding today, after ASCII, is ISO-8859-1. It is an 8-bit superset of ASCII and provides most of the characters necessary for Western Europe.

However, East Asian writing systems needed a way to store over 10,000 characters and so double-byte character sets (DBCS) were developed to provide enough space for the thousands of ideographic characters in East Asian writing systems. Here, the encoding is still byte-based, but each two bytes together represent a single character.

Even in East Asia, text contains letters from small alphabets like Latin or Katakana. These are represented more efficiently with single bytes. Multi-byte character sets (MBCS) provide for this by using a variable number of bytes per character, which distinguishes them from the DBCS encodings. MBCSs are often compatible with ASCII; that is, the Latin letters are represented in such encodings with the same bytes that ASCII uses. Some less often used characters may be encoded using three or even four bytes.

An important feature of MBCSs is that they have byte value ranges that are dedicated for lead bytes and trail bytes. Special ranges for lead bytes, the first bytes in multibyte sequences, make it possible to decide how many bytes belong together to encode a single character. Traditional MBCS encodings are designed so that it is easy to go forwards through a stream of bytes and read characters. However, it is often complicated and very dependent on the properties of the encoding to go backwards in text: going backwards, it is often hard to find out which variable number of bytes represents a single character, and sometimes it is necessary to go forward from the beginning of the text to do this.

Examples of commonly used MBCS encodings are Shift-JIS and EUC-JP (for Japanese), with up to 2 or 3 bytes per character.

Stateful encodings

Some encodings are stateful; they have bytes or byte sequences that switch the meanings of the following bytes. Simple encodings, like mixed-byte EBCDIC, use Shift-In and Shift-Out control characters (bytes) to switch between two states. Sometimes, the bytes after a Shift-In are interpreted as a certain SBCS encoding, and the bytes after a Shift-Out as a certain DBCS encoding. This is very different from an MBCS encoding where the bytes for each character indicate the length of the byte sequence.

The most common stateful encoding is ISO 2022 and its language-specific variations. It uses Escape sequences (byte sequences starting with an ASCII Escape character, byte value 27) to switch between many different embedded encodings. It can also announce encodings that are to be used with special shifting characters in the embedded byte stream. Language-specific variants like ISO-2022-JP limit the set of embeddable encodings and specify only a small set of acceptable Escape sequences for them.

Such encodings are very powerful for data exchange but hard to use in an application. Their flexibility allows you to embed many other encodings, but direct use in programs and conversions to and from other encodings are complicated. For direct use, a program has to keep track not only of the current position in the text, but also of the state--which embeddable encoding is currently active--or must be able to determine the state for a position from considerable context. For conversions to other encodings, converting software might need to have mappings for many embeddable encodings, and for conversions from other encodings, special code must figure out which embeddable encoding to choose for each character.

Why the Unicode Standard?

Hundreds of encodings have been developed, each for small groups of languages and special purposes. As a result, the interpretation of text, input, sorting, display, and storage depends on the knowledge of all the different types of character sets and their encodings. Programs are written to either handle one single encoding at a time and switch between them, or to convert between external and internal encodings.

Part of the problem is that there is no single, authoritative source of precise definitions of many of the encodings and their names. Transferring of text from one machine to another one often causes some loss of information. Also, if a program has the code and the data to perform conversion between a significant subset of traditional encodings, then it carries several megabytes of data around.

The Unicode Standard provides a single character set that covers the languages of the world, and a small number of machine-friendly encoding forms and schemes to fit the needs of existing applications and protocols. It is designed for best interoperability with both ASCII and ISO-8859-1, the most widely used character sets, to make it easier for Unicode to be used in applications and protocols.

The Unicode Standard is in use today, and it is the preferred character set for the Internet, especially for HTML and XML. It is slowly being adopted for use in e-mail, too. Its most attractive property is that it covers all the characters of the world (with exceptions, which will be added in the future). The Unicode Standard makes it possible to access and manipulate characters by unique numbers (that is, their Unicode code points) and use older encodings only for input and output, if at all.

What is z/OS Unicode Services?

z/OS Unicode Services is the Unicode environment on z/OS and consists of two main components:

- z/OS Unicode application programming interfaces services listed below and described in more detail in [Part 2, “Application programmer information,” on page 9](#).
- The infrastructure, described in [Part 3, “System programmer information,” on page 241](#), which provides the z/OS Unicode environment needed to run the z/OS Unicode application programming interfaces.

The z/OS Unicode environment is ready for use after IPL has completed, requiring no action by the system operator.

z/OS support for the Unicode Standard application programming interfaces

z/OS Unicode support is based on Version 13.0 of the Unicode Standard, although lower versions are supported by some services. Review each individual service to see the Unicode Standard versions supported.

z/OS Unicode Services supports the following services:

- Character conversion
- Case conversion
- Normalization
- Collation
- Stringprep
- Bidirectional transformation
- Conversion information service
- Dynamic locale service

Summary information on these services is listed below. For detailed information about these services, see the individual chapters for each service.

Character conversion

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

This support is provided in three ways:

Direct conversion

The conversion from CCSID A to CCSID B is completed with one mapping table that contains all the information needed. z/OS support for the Unicode Standard provides direct conversion between character streams that are encoded with CCSIDs listed in [Appendix C, “Conversion tables supplied with z/OS Unicode Services,” on page 335](#).

Indirect conversion

The conversion from CCSID A to CCSID B is completed in two steps by using Unicode (1200) as the intermediate mapping.

Composite conversion

The conversion of MBCS characters uses several steps to complete the 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 the Unicode Standard. You can find a detailed description of the internal handling in [Appendix B, “Conversion support for multi-byte encodings \(MBCS\),” on page 329](#). An example and an illustration is included as well.

Note: The interface to all three character conversion methods is the same (z/OS Unicode Services uses the indirect or composite method if it is needed).

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 the Unicode Standard must be called in primary mode.

The character conversion service also includes support for bidi transformations.

Case conversion

Case conversion allows conversion to upper, lower, title, or INITCAP case.

z/OS support for the Unicode Standard provides case conversions that allow users to convert the Unicode Standard 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 [The Unicode Consortium \(www.unicode.org\)](http://www.unicode.org).

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 the Unicode Standard 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, see *Unicode Normalization Forms*, which is available in the [Unicode Technical Reports \(www.unicode.org/reports\)](http://www.unicode.org/reports).

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

Collation

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.

z/OS Support for the Unicode Standard 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 Standard collation process, see *Unicode Collation Algorithm* in the [Unicode Technical Reports \(www.unicode.org/reports\)](http://www.unicode.org/reports).

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

Stringprep

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.

z/OS support for the Unicode Standard provides String preparation for internationalized string useful for some internet protocols. This feature is based on RFC 3454. (For more information, see [RFC 3454 \(tools.ietf.org/html/rfc3454\)](http://tools.ietf.org/html/rfc3454).)

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

Bidirectional transformation

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.

Two levels of bidi support are provided:

- The original support, which implements a limited portion of the Unicode Consortium's bidi standard, is available via the bidi service and the character conversion service B technique. These are equivalent functions.

Introduction

- Extended bidi support, which implements more of the Unicode Consortium's bidi standard, is available through the character conversion extended bidi support.

The original bidi transformation service is called using a stub routine named CUNLBIDI for AMODE (31) and CUN4LBID for AMODE (64), or CUNLCNV for AMODE(31) and CUN4LCNV for AMODE(64) if the service B technique is specified.

The extended bidi transformation service is called using a stub routine named CUNLCNV for AMODE (31) and CUN4LCNV for AMODE (64).

Conversion information service

z/OS support for the Unicode Standard 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 CUN4LINF for AMODE (64).

Dynamic locale service

z/OS support for the Unicode Standard provides the dynamic locale service for dynamically building and loading locale data into the z/OS Unicode Services environment to be used by applications with locale-sensitive data.

The dynamic locale service is called using a stub routine called CUNLLOCB for AMODE (31) and CUN4LLOC for AMODE (64).

Part 2. Application programmer information

Chapter 2. About the application programming interfaces

Part 2, “Application programmer information,” on page 9 describes how application programmers are to use the programming interfaces provided by z/OS Unicode Services.

This topic describes some of the key concepts and terminology necessary to understand how to use the Unicode interfaces correctly.

z/OS Unicode environment

The z/OS Unicode environment is an area of the system used to store data needed by z/OS Unicode Services to do its work, such as character conversion tables. It is created during IPL and is accessible from all jobs.

No setup is needed to begin using the z/OS Unicode Services. As of release 1.7, z/OS ships with the z/OS Unicode Services ready to use. An empty z/OS Unicode environment is created, and data is loaded into the environment as needed.

Note: The system programmer can cause conversions to be loaded during IPL if needed.

Application programmers do not work directly with the z/OS Unicode environment. This is because (as of z/OS release 1.7) z/OS Unicode Services automatically loads its resources into the z/OS Unicode environment as needed. This is also referred to as Unicode on-demand or dynamic loading of conversion data.

General concepts when using the z/OS Unicode Services programming interfaces

z/OS Unicode Services provides services in the form of programming interfaces. These are sometimes referred to as application programming interfaces, or APIs. An example of one is the "character conversion service".

Many of these interfaces use the same concepts and field types, such as:

Parameter area

Each programming interface defines a parameter area or an area of storage provided by the caller and used to pass data to the service and to get results back from the service.

Parameter area defaults

Each service defines a constant to initialize the parameter area to default values.

Note: The default value is not necessarily all binary zeroes.

A typical use for the default initializer constant is to initialize the parameter area before changing it to reflect the specific inputs required.

Dynamic Data Area (DDA) required

Some of the services require callers to define a DDA or an area of storage needed and used by the service to perform its function. This storage should be on a word boundary, but does not have to be initialized because it will be modified by the service. The size of the DDA required depends on things such as the parameter area version used, the function selected, and details such as the character data in the source buffer. Most services define a DDA length that is sufficient to accommodate all requests. It is recommended that this length be used.

Parameter area version

Most of the parameter areas define a "version" parameter. The initial version is typically 1 and then incrementally advanced as the parameter area gets larger to accommodate more parameters. The version level controls things such as how big the parameter area is, how much DDA is required,

the functions that are available, and what parameter values are valid. It is recommended that new applications be written to use the latest Unicode Services parameter area version.

ALET support

z/OS Unicode Services interfaces generally allow its DDA and buffers to reside in any address space located by an Access List Entry Token (ALET).

Abstract character data

Abstract character data is a stream of bytes that represent abstract characters. For example, in EBCDIC CCSIDs, the abstract character data bytes x'C9C2D4' represent the abstract characters 'IBM'. Abstract character data is usually referred to as character data or character strings.

Buffers

z/OS Unicode Services that operate on abstract character data have parameters for a source buffer and target buffer. Some services also require a work buffer to store intermediate results. Each buffer is defined by three parameters: a pointer to the buffer, the buffer's ALET, and the buffer's length in bytes.

Note: z/OS Unicode Services typically increment the pointer and decrement the length to indicate how much of the buffer has been used.

Buffer sizes

z/OS Unicode Services that operate on abstract character data have different requirements for target buffer size. The recommended target buffer size is typically a function of the source buffer size and the function requested. For example, when converting from 1-byte Unicode to 2-byte Unicode, the target buffer is typically twice the size of the source buffer. Each API documents its buffer size requirements. The same example applies to the size required for work buffers. Maximum buffer size is limited only by system resources.

Conversion data

Conversion data refers to the data z/OS Unicode Services needs to perform a conversion, such as tables that map from ASCII to EBCDIC. It does not refer to the caller's source buffer. For example, when the character conversion service is called to convert from CCSID 00037 to CCSID 00437, it needs a control block with information about the conversion (information such as both CCSIDs are single-byte) and it needs a 256 byte table to translate the character data. Conversion data is not normally exposed by Unicode Services. The conversion data is stored within the Unicode environment and various interfaces use a 'conversion handle' to refer to conversion data.

Return and reason codes

Generally, z/OS Unicode Services communicate by setting return code and reason code parameters. These values should always be checked when the API returns control.

Note: The parameter area may be left in an inconsistent state when there is a program interrupt.

See [Appendix H, “Unicode return and reason codes,” on page 597](#) for return and reason codes.

Parameters not validated

z/OS Unicode Services do not validate the parameter area before using it. If the parameter area is not filled in properly, unexpected results may occur, including incorrect results, bad return codes, or program interrupts. Unicode Services does not generally monitor for internal errors. The caller is responsible for handling errors.

Page-fixed storage

Callers running with key 0-7 can request that conversion data be loaded into page-fixed storage within the Unicode environment as a way to improve performance by reducing the number of page faults. However, there is no guarantee that the conversion request will result in conversion data being loaded because the conversion data may already be loaded into non-page-fixed storage. To ensure your conversion is loaded into page-fixed storage, you need to work with your system programmer to implement with a PARMLIB member.

Using page-fixed storage is not recommended. There is no guarantee that performance will improve if the conversion is page-fixed.

Note: Callers are free to page-fix the storage they pass into z/OS Unicode Services APIs. Also, z/OS Unicode Services modules themselves are not-page fixed and z/OS Unicode Services is not guaranteed to run with dynamic address translation (DAT) off.

Invoking the z/OS Unicode Services interfaces

z/OS Unicode Services are normally invoked by calling a routine provided by linking a stub routine into application code. These stub routines are located in SYS1.CSSLIB. Some interfaces can be invoked by branching to a control offset, as shown in Appendix G, “System control offsets,” on page 595. This technique may improve performance by eliminating some parameter checking. It is recommended that most customers use the stub routines provided.

Conversion handle use

Some z/OS Unicode Services define a 'conversion handle' parameter. Conversion handles are generated automatically by the conversion service and are available as a way to improve performance.

When a conversion service is invoked, it attempts to locate the conversion data needed:

- If a conversion handle is not provided (for example, it is set to all binary zero), the service resolves the resources needed, then generates a handle to them and stores the handle in the parameter area.
- If a conversion handle is provided, the service checks if the conversion handle is valid. If it is valid, the service does not resolve to the resources specified because it already has this information.

Once the conversion handle is either generated or validated, the service uses it to perform the conversion.

One use of the conversion handle is when you have multiple conversions with the same conversion data and want to optimize performance. For example, when you have multiple buffers that all require the same conversion. Unicode Services lets you re-use conversion handles, saving the effort of re-generating the conversion handle. However, re-using conversion handles requires more from the caller.

The sophisticated usage pattern is:

1. Set the conversion handle to all binary 0.
2. Optional: Invoke the conversion service with an empty source buffer, only to generate the conversion handle. If this step is omitted, the handle will be generated in the next step.
3. Set values into the parameter area, leaving alone the conversion handle. Next, invoke the conversion service and check the return code.
4. Repeat the previous step, making sure to reset any values changed by the conversion service. If you have a different conversion to perform (such as a different source CCSID or target CCSID), also set those values into the parameter area and zero out the conversion handle before repeating the previous step.

Note:

1. If a handle is provided, it is used regardless of the settings of the parameters used to generate it (such as the From CCSID).
2. If the handle needs to be re-generated, the parameter area values will be used to re-generate the handle. It is recommended that you do not modify these key parameters if you are also re-using handles.
3. Handles are invalidated when the Unicode environment changes, such as when adding or deleting a conversion. For example, with the SETUNI DELETE,ALL,FORCE=YES command that may be needed when conversion data is updated via a PTF. Conversion handles are not valid between IPLs of the system. When the conversion service is given an invalid handle, it either returns with an error or generates a valid conversion handle and continues, depending on the setting of the Inv_Handle flag in the Flag1 parameter. It is recommended that most customers set the Inv_Handle flag to 1 to regenerate a new handle.

Sample code

z/OS Unicode Services provides sample source code to invoke the z/OS Unicode Services functions. These are shipped in data set SYS1.SAMPLIB. The API documentation indicates which data set members contain the sample code.

Characteristics for the caller

The programming interfaces share several characteristics, such as:

- z/OS Unicode Services supports the programming languages HLASM, C, and C++. Both 31-bit and 64-bit addressing mode versions of these interfaces are provided.
- They are callable from any key.
- They are callable from problem or supervisor state.
- They are callable in task or SRB mode.
- They are callable in cross-memory mode.
- Header files and sample code are provided.

Linkage conventions

z/OS Unicode Services interfaces follow the MVS linkage conventions described in "Linkage Conventions" of *z/OS MVS Programming: Assembler Services Guide*. The topic for each z/OS Unicode interface gives specific details about the conventions that are followed. In general,

- GPR 1 - Caller must set to the address of the parameter area.
- GPR 13 - Caller must set to the address of a save area.
- GPR 14 - Caller must set to the return address.
- GPR 15 - Caller must set to the entry address. The stub routines do this automatically.

Bidi function

z/OS Unicode Services provides bidirectional and character shaping (bidi) services in two forms:

- Bidi transformation service
- B technique of the character conversion service

The conversions performed are equivalent, except the character conversion service has more options. Bidi conversion options are provided as part of the character conversion service and do not have a separate bidi conversion, so consider using the character conversion interface for new applications.

Related services

Other z/OS components provide some Unicode Standard functions and are not part of the z/OS Unicode Services function, such as:

- Hardware instructions such as "Unpack Unicode" and "Convert UTF-8 to UTF-16".
- C Run-time functions such as iconv.

Chapter 3. Character conversion

This topic describes the character conversion services and the bidirectional and character shaping transformation services (bidi). You can use one or both services with a single API invocation.

The character conversion services convert character data from one representation to another. Character representations are denoted by a coded character set identifier (CCSID) established by the Character Data Representation Architecture (CDRA). For example, z/OS typically stores character data in EBCDIC CCSID 1047.

The bidi transformation services implement some of the specifications described in *Unicode Bidirectional Algorithm*, which is available in the [Unicode Technical Reports \(www.unicode.org/reports\)](http://www.unicode.org/reports). Typically, this service is used to transform character data into a format suitable for display. It does things like changing character data from 'left to right' to 'right to left' and replacing characters with their shaped version. The exact transformation to be performed is specified by fields in the parameter area. Many of these parameters are exactly equivalent to parameters described by the Open Group's portable layout service, which is available at [The Open Group \(opengroup.org\)](http://The Open Group (opengroup.org)).

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.

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

Instead of the CCSIDs, a handle can be given as input. This is possible after the first call because the handle that was used is returned. This helps to speed up the future conversions 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 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).
- Conversion technique (or conversion handle in subsequent calls).
- Work buffer pointer, ALET, and length (see Note 2).
- Dynamic data area pointer (DDA), ALET, and length.
- Flags.

Note:

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 <=> UTF-8 and EBCDIC MBCS <=> UTF-16 conversions, the application developer can provide larger work and target buffers. 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).
- Conversion technique (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.

Note:

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 <=> UTF-8 and EBCDIC MBCS <=> UTF-16 conversions, the application developer can provide larger work and target buffers. 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. Internally, the following conversions are done in the following steps (an indirect conversion):

- A mixed code page and anything other than simple code pages
- UTF-8 and anything other than UTF-16
- A conversion requesting bidi transformations

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 addressing mode (AMODE(31) or AMODE(64)), whether the B technique is requested, and the parameter area version being used. If the 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. It is recommended 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 76.

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 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 28 for the specific values and their definitions.

For the internal handling of MBCS conversions, refer to [Appendix B, “Conversion support for multi-byte encodings \(MBCS\),”](#) on page 329.

Calling the bidi conversion services

This section describes how to use the bidi and character shaping services to transform character data to accommodate bidirectional texts. For example, you can transform text to a form suitable for display.

The bidi transformation works with character conversion. The character conversion service can perform CCSID conversions or CCSID conversions with bidi transformations.

The bidi support operates on CCSID 1200. If the source and target CCSIDs are not both 1200 (or equivalent CCSIDs), the bidi algorithm will cause a two-stage conversion to be performed regardless of any other considerations. The source buffer is first converted to CCSID 1200, bidi transformations are performed, and then the characters are converted to the target CCSID. The work buffer (Wrk_Buf) is required for this.

Note: Use of the extended bidi support for CCSIDs other than Arabic or Hebrew CCSIDs 00420, 00424, 00425, 00856, 00862, 00864, 00916, 01046, 01089, 01255, 01256 will result in a RC = CUN_RC_USER_ERR and a RS = CUN_RS_CCSID_NOT_SUPP.

The bidi support meets some of the standards set forth in the Unicode Consortium's Standard Annex #9. The annex can be found in the [Unicode Technical Reports \(www.unicode.org/reports\)](http://www.unicode.org/reports).

There are two different levels of bidi support:

1. The B technique. This support:

- Is requested by adding the letter B to the technique field.
- Uses parameters Bidi_Context and Bidi_ImpAlg in the Flag1 field in the character conversion parameter area (CUNBCPRM or CUN4BCPR).
- Invokes an older version of the bidi algorithm. This support is equivalent to the bidi transformation service (CUNLBIDI and CUN4LBID).
- Can be used only with parameter area versions 1 and 2.

2. The extended bidi support. This support:

- Is requested by setting the Extended_Bidi_Parm_Area_Ptr.
- Uses parameters in the extended bidi parameter area (CUNBDPRM or CUN4BDPR).
- Invokes a newer version of the bidi algorithm.
- Can be used beginning with parameter area version 3.

IBM does not intend to enhance the B technique support. Instead, it is recommended that you use the extended bidi support for all new development and those who want to use the highest level of bidi support.

If you have code that uses the B technique and you want to change your code to use parameter area version 3, you can no longer use the B technique. In this situation, you must also change your code to use the extended bidi support.

To change your code from the B technique to use the extended bidi support, do the following:

- Use parameter area version 3 and provide the DDA size required for parameter area version 3.
- Set the Extended_Bidi_Parm_Area_Ptr and remove B from the technique letters.
- Instead of setting Flag1 bits Bidi_Context and Bidi_ImpAlg, use fields Context_Src and ImplicitAlg in the extended bidi parameter area.
- Set all other fields in the extended bidi parameter area as appropriate.

Using the extended bidi support requires an extended bidi parameter area.

The extended bidi parameter area is defined by structures CUNBDPRM (for 31-bit service) and CUN4BDPR (for 64-bit service). This parameter area is different than the bidi parameter area defined by the bidi transformation service.

Many of the field names in the extended bidi parameter area specify attributes of either the source or target character string. This is denoted by Src or Targ in the field name. For example, Context_Targ specifies the context for the target string.

Restrictions for the calling environment

Table 1. 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” on page 19. 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     */
```

Character conversion

```

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 Res3;                          /* 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 */
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. */
        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;
    struct {
        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 it is */
            /* not available. */
            /* Note. When conversion are not */
            /* requested from 1200 to */
            /* 1208 and vice versa, the */
            /* contents of this is not */
            /* meaningful. */
    } Flag2;
    unsigned char Designator; /* reserved for ISO 2022 */
    long Return_Code;
    long Reason_Code;
    unsigned int Res6; /* Reserved */
    struct {
        int ETF3E_Behavior : 1, /* Exploit ETF3 HW Enhancement */

```



```

/* between the following */
/* conversions: */
/* 1200->1208 and vice versa */
/* 1200->1232 */
/* 1208->1232 */
/* 0 = Do not exploit ETF3 */
/* HW Enhancement (default) */
/* 1 = Exploit ETF3 */
/* HW Enhancement */
BOM_Removal : 1, /* Try to remove BOM if */
/* source CCSID is Unicode */
/* 0 = Do not remove BOM */
/* (default) */
/* 1 = Remove BOM if it exists */
ISO2022_JP_ESC : 1, /* Try to add ISO2022-JP Roman */
/* or ASCII ESC sequence at the */
/* beginning of target buffer. */
/* 0 = Do not add Roman or */
/* ASCII ESC sequence */
/* (default) */
/* 1 = Add Roman or */
/* ASCII ESC sequence */
Flag3Res : 13; /* Reserved */
/* FLAG3 Byte 2 set by caller */
/* Reserved */
} Flag3;
char Res7[2];
CUNBDPRM * Extended_Bidi_Parm_Area_Ptr;
char Res8[64];
} CUNBCPRM;

/* The extended bidi parameter area */
typedef struct tagCUNBDPRM {
int Version;
int Length;
struct {
int XOpen_Defaults : 1,
KBS_Defaults : 1,
Keyword : 1,
From_wtransform : 1,
: 4,
} InFlags;
struct {
int Layout_Roundtrip : 1,
Layout_WinCompat : 1,
Layout_ImpToImp : 1,
Layout_Remove_Marks : 1,
Layout_Insert_Marks : 1,
Layout_Streaming : 1,
: 2;
} Layout_Options;
struct {
int ActiveShapeEditing : 1,
ActiveDirectional : 1,
: 14;
} OutFlags;
int Orientation_Src;
int Orientation_Targ;
int Context_Src;
int Context_Targ;
int TypeOfText_Src;
int TypeOfText_Targ;
int ImplicitAlg_Src;
int ImplicitAlg_Targ;
int Swapping_Src;
int Swapping_Targ;
int Numerals_Src;
int Numerals_Targ;
int TextShaping_Src;
int TextShaping_Targ;
int ShapeCharsetSize;
int ShapeContextSize_Front;
int ShapeContextSize_Back;
int CheckMode;
unsigned int InpBufIndex;
unsigned long Streaming_Processed_Length;
int ArabicOneCellShaping_Src;
int ArabicOneCellShaping_Targ;
int WordBreak_Src;
int WordBreak_Targ;
int LamAlefEditMode_Src;
int LamAlefEditMode_Targ;
int YehHamzaMode_Src;
int YehHamzaMode_Targ;

```

Character conversion

```
int      TailEditMode_Src;
int      TailEditMode_Targ;
int      TashkeelEditMode_Src;
int      TashkeelEditMode_Targ;
unsigned int * InpToOut_Ptr;
unsigned int * OutToInp_Ptr;
unsigned char * BidiLvl_Ptr;
char      Layout_Streaming_State[64];
char      Bidi_Keyword[128];
char      Res2[64];
} CUNBDPRM;
```

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. */
        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. */
        int      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;
int Res4;
long Res5; /* Reserved */
struct {
    int ETF3E_Behavior : 1, /* Exploit ETF3 HW */
        /* Enhancement between the */
        /* following conversions: */
        /* 1200->1208 and vice versa */
        /* 1200->1232 */
        /* 1208->1232 */
        /* 0 = Do not exploit ETF3 */
        /* HW Enhancement (default) */
        /* 1 = Exploit ETF3 */
        /* HW Enhancement */
}

BOM_Removal : 1, /* Try to remove BOM if */
    /* source CCSID is Unicode */
    /* 0 = Do not remove BOM */
    /* (default) */
    /* 1 = Remove BOM if it exists */
ISO2022_JP_ESC : 1, /* Try to add ISO2022-JP Roman */
    /* or ASCII ESC sequence at the*/
    /* beginning of target buffer. */
    /* 0 = Do not add Roman or */
    /* ASCII ESC sequence */
    /* (default) */
    /* 1 = Add Roman or */
    /* ASCII ESC sequence */
Flag3Res : 13; /* Reserved */
Flag3; /* FLAG3 Byte 2 set by caller */
char Res7[2]; /* Reserved */
CUN4BDPR * Extended_Bidi_Parm_Area_Ptr;
char Res8[64];
} CUN4BCPR;

typedef struct CUN4BDPR {
    int Version;
    int Length;
    struct {
        int XOpen_Defaults : 1,
            KBS_Defaults : 1,
            Keyword : 1,
            From_wtransform : 1,
            : 4;
    }
    InFlags;
    struct {
        int Layout_Roundtrip : 1,
            Layout_WinCompat : 1,
            Layout_ImpToImp : 1,
            Layout_Remove_Marks : 1,
            Layout_Insert_Marks : 1,
            Layout_Streaming : 1,
            : 2;
    }
    Layout_Options;
    struct {
        int ActiveShapeEditing : 1,
            ActiveDirectional : 1,
            : 14;
    }
    OutFlags;
    char Res1[4];
    int Orientation_Src;
    int Orientation_Targ;

```

```

int          Context_Src;
int          Context_Targ;
int          TypeOfText_Src;
int          TypeOfText_Targ;
int          ImplicitAlg_Src;
int          ImplicitAlg_Targ;
int          Swapping_Src;
int          Swapping_Targ;
int          Numerals_Src;
int          Numerals_Targ;
int          TextShaping_Src;
int          TextShaping_Targ;
int          ShapeCharsetSize;
int          ShapeContextSize_Front;
int          ShapeContextSize_Back;
int          CheckMode;
unsigned long InpBufIndex;
unsigned long Streaming_Processed_Length;
int          ArabicOneCellShaping_Src;
int          ArabicOneCellShaping_Targ;
int          WordBreak_Src;
int          WordBreak_Targ;
int          LamAlefEditMode_Src;
int          LamAlefEditMode_Targ;
int          YehHamzaMode_Src;
int          YehHamzaMode_Targ;
int          TailEditMode_Src;
int          TailEditMode_Targ;
int          TashkeelEditMode_Src;
int          TashkeelEditMode_Targ;
unsigned int * InpToOut_Ptr;
unsigned int * OutToInp_Ptr;
unsigned char * BidiLvl_Ptr;
char          Layout_Streaming_State[64];
char          Bidi_Keyword[128];
char          Res2[64];
} CUN4BDPR;

```

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_CCSID,=FL4'1047'  From CCSID
*      MVC   CUNBCPRM_TARG_CCSID,=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_CCSID,=FL4'1047' From CCSID
*      MVC   CUN4BCPR_TARG_CCSID,=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 2. Mapping of parameters in HLASM for character conversion AMODE (31) | | | | | | |
|---|------------|-----------|-----------------|----------|------------------------|---|
| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
| 0 | (0) | STRUCTURE | 176 | DWORD | CUNBCPRM | Parameter area |
| 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) |
| | | UNSIGNED | 4 | | CUNBCPRM_Targ_CCSID | Target CCSID (codepage) |
| | | 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 |

Table 2. Mapping of parameters in HLASM for character conversion AMODE (31) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|------------------------|-----------------|----------|---------------------------|---|
| 136 | (88) | CHARACTER | 4 | | * | Reserved for 64 bit |
| 140 | (8C) | ADDRESS | 4 | DWORD | 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 |
| 152 | (98) | BITSTRING | 1 | | CUNBCPRM_Flag1 | FLAG Byte 1 set by caller |
| 152 | (98) | 1... .. | 1 | | CUNBCPRM_Sub_Action | Sub action: 0=TERMINATE WITH ERROR 1=Substitute AND CONT |
| 152 | (98) | .1... .. | 1 | | CUNBCPRM_Inv_Handle | Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONT |
| 152 | (98) | ..1. | 1 | | CUNBCPRM_No_Opt_Buf_Fill | Target buffer filled: 0=TARGET BUFFER FILLED OPTIMALLY 1=TARGET BUFFER NOT FILLED OPTIMALLY |
| 152 | (98) | ...1 | 1 | | CUNBCPRM_Mal_Action | Mal Action: (Default 0): 0=SUBSTITUTE AND CONT 1=TERMINATE WITH ERROR |
| 152 | (98) | 1... | 1 | | CUNBCPRM_RL_Sub_Action | R or L technique action |
| 152 | (98) |1.. | 1 | | CUNBCPRM_SrcSub_Chk | Substitution Chars Check in source: 0=Do nothing 1=Override SUB_ACTION |
| 152 | (98) |1. | 1 | | CUNBCPRM_Bidi_Context | Bidi Context: 0=Context LTR 1=Context RTL |
| 152 | (98) |1 | 1 | | CUNBCPRM_Bidi_ImpAlg | Bidi Implicit Alg: 0=Algor Basic 1=Algor Implicit |
| 153 | (99) | UNSIGNED | 1 | | CUNBCPRM_Subcodepage | Number of subcodepage(s) |
| 153 | (99) | BITSTRING 1111 | 1 | | CUNBCPRM_Source_SCP_State | Source subcodepage status |

Table 2. Mapping of parameters in HLASM for character conversion AMODE (31) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|------------------------|-----------------|----------|--------------------------------|--|
| 153 | (99) | BITSTRING 1111 | 1 | | CUNBCPRM_Target_SCP_State | Target subcodepage status |
| 154 | (9A) | BITSTRING | 1 | | CUNBCPRM_Flag2 | FLAG Byte 2 set by service |
| 154 | (9A) | 1... .. | 1 | | CUNBCPRM_Substitution | Substitution: 0=NO CHARACTER SUBSTITUTED 1=CHARACTER(S) SUBSTITUTED. |
| 154 | (9A) | .1... .. | 1 | | CUNBCPRM_Mal_Found | Malformed String found: 0=NO MALFORMED STRING FOUND 1=MALFORMED STRING FOUND. |
| 154 | (9A) | .1. | 1 | | CUNBCPRM_Page_Fix | Page fixing: 0=System storage 1=Page Fixing |
| 154 | (9A) | ...1 | 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) | CHARACTER | 4 | | CUNBCPRM_Subs_Counter | Reserved |
| 168 | (A8) | BITSTRING | 2 | | CUNBCPRM_Flag3 | Flag 3 |
| 168 | (A8) | 1... .. | 2 | | CUNBCPRM ETF3E_Behavior | ETF3 hardware enhancement implementation for conversions from 1200 to 1208 and vice versa: <ul style="list-style-type: none">0=Do not exploit ETF3 hardware enhancement.1=Exploit ETF3 hardware enhancement. |
| 168 | (A8) | .1... .. | 2 | | CUNBCPRM_BOM_Removal | Tries to remove BOM if source CCSID is Unicode: <ul style="list-style-type: none">0 = Do not remove BOM (default).1 = Remove BOM if it exists. |

Table 2. Mapping of parameters in HLASM for character conversion AMODE (31) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|-----------|-----------------|----------|--------------------------------------|--|
| 168 | (A8) | ..1. | 2 | | CUNBCPRM_ISO2022_JP_ESC | Tries to add ISO2022-JP Roman or ASCII ESC sequence at the beginning of target buffer. <ul style="list-style-type: none"> 0 = Do not add Roman or ASCII ESC sequence (default). 1 = Add Roman or ASCII ESC sequence. |
| 170 | (AA) | CHARACTER | 2 | | * | Reserved |
| 172 | (AC) | ADDRESS | 4 | | CUNBCPRM_Extended_Bidi_Parm_Area_Ptr | Points to the bidi parm area |
| 176 | (B0) | | 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.

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

Parameter value CUNBCPRM_Version3 is defined for extended bidi support.

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'7FFFFFFF'.

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'7FFFFFFF'.

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 256 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 256 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 z/OS Unicode Services loads conversion tables” on page 264. In addition to the techniques search orders (R,E,C,L,M,P 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:

- The B technique can be combined in any order with the current supported techniques search orders (R,E,C,L,M,P, and 0-9).
- When the B technique is requested, CUNBCPRM_DDA_Req2 must be used as DDA value for CUNBCPRM_DDA_Buf_Len.
- The B technique is not supported by the Image generator CUNMIUTL.
- The B technique is not part of the default technique search order RECLM.
- The B technique is not supported through the SETUNI command.
- The B technique can only be used with parameter area version 1 or 2.

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 15](#). 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. The required length depends on the type of conversion being done (source and target CCSIDs), the addressing mode (AMODE(31) or AMODE(64)), whether the B technique is requested, and the parameter area version being used.

The following recommendations are for all conversion types:

- For parameter area version 1 or 2, use CUNBCPRM_DDA_Required. When the B technique is used (with parameter area version 1 or 2), use CUNBCPRM_DDA_Req2.
- For parameter area version 3, use CUNBCPRM_DDA_Req3.
- For AMODE(64), use the CUN4BCPR versions of the constants.

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_Mal_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. This field is for the B technique.

- **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. This field is for the B technique.

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

For more information, see [Chapter 7, “Bidi transformation,” on page 169](#).

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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded into page-fixed memory.

- **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.

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 |
| x1xx xxxx | CUNBCPRM_BOM_Removal |
| xx1x xxxx | CUNBCPRM_ISO2022_JP_ESC |

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.

CUNBCPRM_BOM_Removal

Specifies whether to remove BOM from conversion from Unicode to other CCSIDs.

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 remove BOM. 0 is the default.
- **1:** Remove BOM if possible.

CUNBCPRM_ISO2022_JP_ESC

Specifies whether to add ISO2022-JP Roman or ASCII ESC sequence at the beginning of target buffer.

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 add Roman or ASCII ESC sequence. 0 is the default.
- **1:** Add Roman or ASCII ESC sequence.

CUNBCPRM_Extended_Bidi_Parm_Area_Ptr - set by caller

Optionally specifies the address of the extended bidirectional and character shaping parameter area. This parameter area must be in the primary address space. The parameter area must be aligned on a doubleword boundary. Use a zero pointer value to indicate that the bidi and character shaping service is not to be used.

This field was added in parameter area version 3.

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 3. 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 | 216 | DWORD | CUN4BCPR | Parameter area |
| 0 | (0) | UNSIGNED | 4 | | CUN4BCPR_Version | Parameter area VERSION |
| 4 | (4) | UNSIGNED | 4 | | CUN4BCPR_Length | Parameter area Length |
| 8 | (8) | ADDRESS | 8 | | CUN4BCPR_Src_Buf_Ptr | Source buffer pointer |
| 16 | (10) | UNSIGNED | 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) | ADDRESS | 8 | | CUN4BCPR_Targ_Buf_Ptr | Target buffer pointer |
| 40 | (28) | UNSIGNED | 4 | | CUN4BCPR_Targ_Buf_ALET | Target buffer ALET |
| 44 | (2C) | UNSIGNED | 4 | | * | Reserved |

Table 3. 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 |
|------------|------------|-----------|-----------------|----------|--------------------------|---|
| 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) |
| 120 | (78) | UNSIGNED | 4 | | CUN4BCPR_Targ_CCSID | Target CCSID (codepage) |
| 120 | (78) | CHARACTER | 8 | | CUN4BCPR_Technique | The CONVERSION TECHNIQUE is specified as input to the image generator |
| 136 | (88) | ADDRESS | 8 | | CUN4BCPR_Wrk_Buf_Ptr | Work buffer pointer |
| 144 | (90) | UNSIGNED | 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 | (A0) | ADDRESS | 8 | DWORD | 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 |
| 176 | (B0) | BITSTRING | 1 | | CUN4BCPR_Flag1 | FLAG Byte 1 set by caller |
| 176 | (B0) | 1... .. | 1 | | CUN4BCPR_Sub_Action | Sub action: 0=TERMINATE WITH ERROR. 1=Substitute AND CONT. |
| 176 | (B0) | .1... .. | 1 | | CUN4BCPR_Inv_Handle | Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONT. |
| 176 | (B0) | ..1. | 1 | | CUN4BCPR_No_Opt_Buf_Fill | Target buffer filled: 0=TARGET BUFFER FILLED OPTIMALLY. 1=TARGET BUFFER NOT FILLED OPTIMALLY. |
| 176 | (B0) | ...1 | 1 | | CUN4BCPR_Mal_Action | Mal Action: (Default 0): 0=SUBSTITUTE AND CONT. 1=TERMINATE WITH ERROR |
| 176 | (B0) | 1... | 1 | | CUN4BCPR_RL_Sub_Action | R or L technique action |

Table 3. 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) |1.. | 1 | | CUN4BCPR_SrcSub_Chk | Substitution Chars Check in source: 0=Does nothing. 1=Override SUB_ACTION. |
| 176 | (B0) |1. | 1 | | CUN4BCPR_Bidi_Context | Bidi Context: 0=Context LTR 1=Context RTL |
| 176 | (B0) |1 | 1 | | CUN4BCPR_Bidi_ImpAlg | Bidi Implicit Alg: 0=Algor Basic 1=Algor Implicit |
| 177 | (B1) | UNSIGNED | 1 | | CUN4BCPR_Subcodepage | Number of subcodepage(s) |
| 177 | (B1) | BITSTRING 1111 | 1 | | CUN4BCPR_Source_SCP_State | Source subcodepage status |
| 177 | (B1) | BITSTRING 1111 | 1 | | CUN4BCPR_Target_SCP_State | Target subcodepage status |
| 178 | (B2) | BITSTRING | 1 | | CUN4BCPR_Flag2 | FLAG Byte 2 set by service |
| 178 | (B2) | 1... | 1 | | CUN4BCPR_Substitution | Substitution: 0=NO CHARACTER SUBSTITUTED. 1=CHARACTER(S) SUBSTITUTED. |
| 178 | (B2) | .1... | 1 | | CUN4BCPR_Mal_Found | Malformed string found: 0=NO MALFORMED STRING FOUND 1=MALFORMED STRING FOUND. |
| 178 | (B2) | ..1. | 1 | | CUN4BCPR_Page_Fix | Page fixing: 0=System storage 1=Page Fixing |
| 178 | (B2) | ...1 ... | 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 |

| Table 3. 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 |
| 180 | (B4) | UNSIGNED | 4 | | CUN4BCPR_Return_Code | Return code |
| 184 | (B8) | UNSIGNED | 4 | | CUN4BCPR_Reason_Code | Reason code |
| 188 | (BC) | UNSIGNED | 4 | | * | Reserved |
| 192 | (C0) | CHARACTER | 8 | | CUN4BCPR_Subs_Counter | Reserved |
| 200 | (C8) | BITSTRING | 2 | | CUN4BCPR_Flag3 | Flag 3 |
| 200 | (C8) | 1... .. | 2 | | 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. |
| 200 | (C8) | .1... .. | 2 | | CUN4BCPR_BOM_Removal | Tries to remove BOM if source CCSID is Unicode. <ul style="list-style-type: none">0 = Do not remove BOM (default).1 = Remove BOM if it exists. |
| 200 | (C8) | ..1. | 2 | | CUN4BCPR_ISO2022_JP_ESC | Tries to add ISO2022-JP Roman or ASCII ESC sequence at the beginning of target buffer. <ul style="list-style-type: none">0 = Do not add Roman or ASCII ESC sequence (default).1 = Add Roman or ASCII ESC sequence. |
| 202 | (CA) | UNSIGNED | 6 | | * | Reserved |
| 208 | (D0) | ADDRESS | 8 | | CUN4BCPR_Extended_Bidi_Parm_Area_Ptr | Points to the bidi parm area |
| 216 | (D8) | | 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.

Parameter value CUN4BCPR_Version2 is defined to exploit the ETF3 hardware enhancement function.

Parameter value CUN4BCPR_Version3 is defined for extended bidi support.

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 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 256](#) 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 CUNLCNV. 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 256](#) 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 263](#). In addition to the techniques search orders (R,E,C,L,M,P and 0-9) that are supported currently, you can also use the B technique to invoke bidi service through the character conversion service API. When the B technique is requested, target buffer will contain the to-CCSID conversion plus bidi properties. Consider the following characteristics when you use the B technique:

- The B technique can be combined in any order with the current supported techniques search orders (R,E,C,L,M,P and 0-9).
- When The B technique is requested, CUN4BCPR_DDA_Req2 must be used as DDA value for CUN4BCPR_DDA_Buf_Len.
- The B technique is not supported by the Image generator CUNMIUTL.
- The B technique is not part of default technique search order RECLM.
- The B technique is not supported through the SETUNI command.
- The B technique can only be used with parameter area version 1 or 2.

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 15](#). 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. The required length depends on the type of conversion being done (source and target CCSIDs), the addressing mode (AMODE(31) or AMODE(64)), whether the B technique is requested, and the parameter area version being used.

The following recommendations are for all conversion types:

- For parameter area version 1 or 2, use CUN4BCPR_DDA_Required. When the B technique is used (with parameter area version 1 or 2), use CUN4BCPR_DDA_Req2.
- For parameter area version 3, use CUN4BCPR_DDA_Req3.
- For AMODE(31), use the CUNBCPRM versions of the constants.

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. This field is for the B technique.

- **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. This field is for the B technique.

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

For more information, see [Chapter 7, “Bidi transformation,” on page 169](#).

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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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 |
| x1xx xxxx | CUN4BCPR_BOM_Removal |
| xx1x xxxx | CUN4BCPR_ISO2022_JP_ESC |

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.

CUN4BCPR_BOM_Removal

Specify whether to remove BOM from conversion from Unicode to other CCSIDs.

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 remove BOM. 0 is the default.
- **1:** Remove BOM if possible.

CUN4BCPR_ISO2022_JP_ESC

Specifies whether to add ISO2022-JP Roman or ASCII ESC sequence at the beginning of target buffer.

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 add Roman or ASCII ESC sequence. 0 is the default.
- **1:** Add Roman or ASCII ESC sequence.

CUN4BCPR_Extended_Bidi_Parm_Area_Ptr - set by caller

Optionally specifies the address of the extended bidirectional and character shaping parameter area. This parameter area must be in the primary address space. The parameter area must be aligned on a doubleword boundary. Use a zero pointer value to indicate that the bidi and character shaping service is not to be used.

This field was added in parameter area version 3.

Mapping of the extended bidi parameter area

The HLASM mapping of the extended bidi parameter area is given in interface definition files CUNBCIDF (for 31-bit) and CUN4BCID (for 64-bit) in dataset SYS1.MACLIB.

AMODE(31)

Table 4. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(31)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|------------------------|-----------------|----------|------------------------------|---|
| 0 | (0) | STRUCTURE | | DWORD | CUNBDPRM | Extended bidi parameter area for 31-bit character conversion |
| | (0) | UNSIGNED | 4 | | CUNBDPRM_Version | Version of the parameter area |
| | (4) | UNSIGNED | 4 | | CUNBDPRM_Len | Length, in bytes, of the parameter area |
| | (8) | BITSTRING | 1 | | CUNBDPRM_InFlags | Input flags |
| | | 1... | 1 | | CUNBDPRM_XOpen_Defaults | Specifies X/Open portable layout option defaults |
| | | .1.. | 1 | | CUNBDPRM_KBS_Defaults | Specifies Unicode Services knowledge base defaults |
| | | ..1. | 1 | | CUNBDPRM_Keyword | Specifies bidi keyword |
| | | ...1 | 1 | | CUNBDPRM_From_wtransform | Reserved for Unicode Services use. This should not be set by users. |
| | (9) | BITSTRING | 1 | | CUNBDPRM_Layout_Options | Layout options |
| | | 1... | 1 | | CUNBDPRM_Layout_Roundtrip | Specifies if round trip processing is to be used |
| | | .1.. | 1 | | CUNBDPRM_Layout_WinCompat | Specifies if the WinCompat mode is to be used |
| | | ..1. | 1 | | CUNBDPRM_Layout_ImpToImp | Specifies if a 'logical to logical' transformation is to be performed |
| | | ...1 | 1 | | CUNBDPRM_Layout_Remove_Marks | Specifies if all bidi marks will be removed |
| | | 1... | 1 | | CUNBDPRM_Layout_Insert_Marks | Specifies if bidi marks are to be inserted |
| | |1.. | 1 | | CUNBDPRM_Layout_Streaming | Specifies if layout streaming is to be used |
| | (A) | BITSTRING | 2 | | CUNBDPRM_OutFlags | Output flags |
| | | 1... | | | CUNBDPRM_ActiveDirectional | Specifies if directional elements were used |
| | | .1.. | | | CUNBDPRM_ActiveShapeEditing | Specifies if caller must perform shape editing |
| | (C) | CHARACTER | 4 | | Reserved | |
| | (10) | UNSIGNED | 4 | | CUNBDPRM_Orientation_Src | Orientation of the source buffer |
| | (14) | UNSIGNED | 4 | | CUNBDPRM_Orientation_Targ | Orientation of the target buffer |
| | (18) | UNSIGNED | 4 | | CUNBDPRM_Context_Src | Context of the source buffer |
| | (1C) | UNSIGNED | 4 | | CUNBDPRM_Context_Targ | Context of the target buffer |
| | (20) | UNSIGNED | 4 | | CUNBDPRM_TypeOfText_Src | Type of text of the source buffer |

Table 4. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(31) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|----------|-----------------|----------|-------------------------------------|--|
| | (24) | UNSIGNED | 4 | | CUNBDPRM_TypeOfText_Targ | Type of text of the target buffer |
| | (28) | UNSIGNED | 4 | | CUNBDPRM_ImplicitAlg_Src | Implicit algorithm used in the source buffer |
| | (2C) | UNSIGNED | 4 | | CUNBDPRM_ImplicitAlg_Targ | Implicit algorithm used in the target buffer |
| | (30) | UNSIGNED | 4 | | CUNBDPRM_Swapping_Src | Swapping used in the source buffer |
| | (34) | UNSIGNED | 4 | | CUNBDPRM_Swapping_Targ | Swapping used in the target buffer |
| | (38) | UNSIGNED | 4 | | CUNBDPRM_Numerals_Src | Numerals used in the source buffer |
| | (3C) | UNSIGNED | 4 | | CUNBDPRM_Numerals_Targ | Numerals used in the target buffer |
| | (40) | UNSIGNED | 4 | | CUNBDPRM_TextShaping_Src | Text shaping used in the source buffer |
| | (44) | UNSIGNED | 4 | | CUNBDPRM_TextShaping_Targ | Text shaping used in the target buffer |
| | (48) | UNSIGNED | 4 | | CUNBDPRM_ShapeCharsetSize | Size of elements of the character set |
| | (4C) | UNSIGNED | 4 | | CUNBDPRM_ShapeContextSize_Front | Number of code elements required for shape editing |
| | (50) | UNSIGNED | 4 | | CUNBDPRM_ShapeContextSize_Back | Number of code elements required for shape editing |
| | (54) | UNSIGNED | 4 | | CUNBDPRM_CheckMode | Level of bidi checking |
| | (58) | UNSIGNED | 4 | | CUNBDPRM_InpBufIndex | Bidi input buffer index |
| | (5C) | UNSIGNED | 4 | | CUNBDPRM_Streaming_Processed_Length | Bidi streaming processed length |
| | (60) | UNSIGNED | 4 | | CUNBDPRM_ArabicOneCellShaping_Src | Arabic one-cell shaping used in the source buffer |
| | (64) | UNSIGNED | 4 | | CUNBDPRM_ArabicOneCellShaping_Targ | Arabic one-cell shaping used in the target buffer |
| | (68) | UNSIGNED | 4 | | CUNBDPRM_WordBreak_Src | Word break used in the source buffer |
| | (6C) | UNSIGNED | 4 | | CUNBDPRM_WordBreak_Targ | Word break used in the target buffer |
| | (70) | UNSIGNED | 4 | | CUNBDPRM_LamAlefEditMode_Src | LamAlef edit mode used in the source buffer |
| | (74) | UNSIGNED | 4 | | CUNBDPRM_LamAlefEditMode_Targ | LamAlef edit mode used in the target buffer |
| | (78) | UNSIGNED | 4 | | CUNBDPRM_YehHamzaMode_Src | YehHamza edit mode used in the source buffer |
| | (7C) | UNSIGNED | 4 | | CUNBDPRM_YehHamzaMode_Targ | YehHamza edit mode used in the target buffer |
| | (80) | UNSIGNED | 4 | | CUNBDPRM_TailEditMode_Src | Tail edit mode used in the source buffer |

Table 4. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(31) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|-----------|-----------------|----------|---------------------------------|--|
| | (84) | UNSIGNED | 4 | | CUNBDPRM_TailEditMode_Targ | Tail edit mode used in the target buffer |
| | (88) | UNSIGNED | 4 | | CUNBDPRM_TashkeelEditMode_Src | Tashkeel edit mode used in the source buffer |
| | (8C) | UNSIGNED | 4 | | CUNBDPRM_TashkeelEditMode_Targ | Tashkeel edit mode used in the target buffer |
| | (90) | ADDRESS | 4 | | CUNBDPRM_InpToOut_Ptr | Bidi input to output buffer pointer |
| | (94) | ADDRESS | 4 | | CUNBDPRM_OutToInp_Ptr | Bidi output to input buffer pointer |
| | (98) | ADDRESS | 4 | | CUNBDPRM_BidiLvl_Ptr | BidiLvl property pointer |
| | (9C) | CHARACTER | 64 | | CUNBDPRM_Layout_Streaming_State | State of the layout streaming operation |
| | (DC) | CHARACTER | 128 | | CUNBDPRM_Bidi_Keyword | Short form keyword |
| | (15C) | CHARACTER | 64 | | * | Reserved |
| | (19C) | | 0 | | CUNBDPRM_End | End of CUNBDPRM |

Description of parameters in area CUNBDPRM

This description applies to HLASM.

CUNBDPRM_Version - set by caller

Specifies the version of the parameter area. Use version 1.

CUNBDPRM_Length - set by caller

Specifies the length of the parameter area, in bytes. Use constant CUNBDPRM_Len.

CUNBDPRM_InFlags - set by caller (except for CUNBDPRM_From_wtransform)

| Bit position | Name |
|--------------|--------------------------|
| 1xxx xxxx | CUNBDPRM_XOpen_Defaults |
| x1xx xxxx | CUNBDPRM_KBS_Defaults |
| xx1x xxxx | CUNBDPRM_Keyword |
| xxx1 xxxx | CUNBDPRM_From_wtransform |

CUNBDPRM_XOpen_Defaults - set by caller

Specifies whether or not to use default settings for the X/Open portable layout options. Possible values are:

- **0**: Do not use default settings for the X/Open portable layout options.
- **1**: Use default settings for the X/Open portable layout options.

Note: The settings defined in the short-form keyword CUNBDPRM_Bidi_Keyword have higher priority over the defaults. The attributes specified in the bidi keyword will overlay the default attributes.

CUNBDPRM_KBS_Defaults - set by caller

Specifies whether or not to use default settings from the Unicode Services knowledge base to set the X/Open portable layout options. Possible values are:

- **0:** Do not use default settings from the Unicode Services knowledge base to set the X/Open portable layout options.
- **1:** Use default settings from the Unicode Services knowledge base to set the X/Open portable layout options.

Note: This flag is ignored if CUNBDPRM_XOpen_Defaults is ON. If CUNBDPRM_XOpen_Defaults is OFF and CUNBDPRM_KBS_Defaults is ON, the defaults defined in the Unicode Services knowledge base will be used. The bidi string types and associated attributes defined in the knowledge base are based on the input or output CCSID. The settings defined in the short-form keyword CUNBDPRM_Bidi_Keyword have higher priority over the default attributes.

CUNBDPRM_Keyword - set by caller

Specifies whether or not to use the short form keyword to set the X/Open portable layout options. Possible values are:

- **0:** Do not use the short form keyword to set the X/Open portable layout options.
- **1:** Use the short form keyword to set the X/Open portable layout options.

Note: This flag must be set to ON when the CUNBDPRM_Bidi_Keyword is used.

CUNBDPRM_From_wtransform - set by service

This flag is reserved for internal Unicode Services use. It should not be set by the caller.

CUNBDPRM_Layout_Options - set by caller

| Bit position | Name |
|--------------|------------------------------|
| 1xxx xxxx | CUNBDPRM_Layout_Roundtrip |
| x1xx xxxx | CUNBDPRM_Layout_WinCompat |
| xx1x xxxx | CUNBDPRM_Layout_ImpToImp |
| xxx1 xxxx | CUNBDPRM_Layout_Remove_Marks |
| xxxx 1xxx | CUNBDPRM_Layout_Insert_Marks |
| xxxx x1xx | CUNBDPRM_Layout_Streaming |

CUNBDPRM_Layout_Roundtrip - set by caller

Specifies if numbers located between LTR text and RTL text are associated with the RTL text. This makes the algorithm reversible and makes it useful when round trip (from visual to logical and back to visual) must be achieved without adding LRM characters. However, this is a variation from the standard Unicode bidi algorithm. Possible values are:

- **0:** Numbers are not associated with the RTL text.
- **1:** Numbers are associated with the RTL text.

CUNBDPRM_Layout_WinCompat - set by caller

Specifies if the algorithm used to perform bidi transformations should approximate the algorithm used in Microsoft Windows XP, rather than strictly conform to the Unicode bidi algorithm. Possible values are:

- **0**: Do not approximate the Microsoft algorithm.
- **1**: Approximate the Microsoft algorithm.

CUNBDPRM_Layout_ImpToImp - set by caller

Specifies if a logical to logical transformation is to be performed:

- If the source orientation is LTR, the source text will be handled as LTR logical text and will be transformed to the RTL logical text which has the same LTR visual display.
- If the source orientation is RTL, the source text will be handled as RTL logical text and will be transformed to the LTR logical text which has the same LTR visual display.

This mode may be needed when logical text, which is basically Arabic or Hebrew, with possible included numbers or phrases in English, has to be displayed as if it had LTR orientation. This can happen if the displaying application treats all text as if it was basically LTR. This mode may also be needed in the reverse case, when logical text which is basically English, with possible included phrases in Arabic or Hebrew, has to be displayed as if it had RTL orientation. The problem may be handled by transforming the source text with this option before displaying it, so that it will be displayed properly. Possible values are:

- **0**: Logical to logical transformation is not to be performed.
- **1**: Logical to logical transformation is to be performed.

CUNBDPRM_Layout_Remove_Marks - set by caller

Specifies if all bidi marks (LRM or RLM) will be removed from the output text when performing a transformation. Possible values are:

- **0**: Bidi marks are not to be removed from the output text.
- **1**: Bidi marks are to be removed from the output text. The corresponding entries in the InpToOut map are set equal to the maximum value. This option should not be specified together with option Layout_Insert_Marks, and it overrides it.

CUNBDPRM_Layout_Insert_Marks - set by caller

Specifies if bidi marks (LRM or RLM) are to be inserted when needed to ensure correct results when reordering to an implicit order. This option is meaningful only when performing a transformation from visually ordered to implicitly ordered text. Possible values are:

- **0**: Do not insert bidi marks.
- **1**: Insert bidi marks. A minimum number of LRM or RLM characters will be added to the source text after reordering it so as to ensure round trip. For example, when applying the inverse transformation on the resulting implicit text with removal of bidi marks (option Layout_Remove_Marks), the result will be identical to the source text in the first transformation. The LRM and RLM characters, which are added in the output text, have no matching character in the source text. The corresponding entries in the OutToInp map are set equal to the maximum value.

Set by caller. Ignored if specified together with CUNBDPRM_Layout_Remove_Marks.

CUNBDPRM_Layout_Streaming - set by caller

Specifies if the caller is interested in using layout streaming. Layout streaming processes large text objects into parts using the piece by piece technique. The results of the successive calls are expected to be concatenated by the caller. Only the call for the last part will have this option bit off. Possible values are:

- **0**: Do not use layout streaming.
- **1**: Attempt to use layout streaming. The transform operation may process less than the full source text in order to truncate the text at a meaningful boundary. The caller must read the value in CUNBDPRM_Streaming_Processed_Length immediately after performing the transform in order to determine how much of the source text has been processed. Source text beyond that length should be resubmitted in following transform operations. If the last character of the source text constitutes a reasonable boundary, the whole text will be processed at once. If no

where in the source text there exists such a reasonable boundary, the processed length will be zero. The caller should check for such an occurrence and do one of the following:

- Submit a larger amount of text with a better chance to include a reasonable boundary.
- Resubmit the same text after turning off this option.

In all cases, this option should be turned off before processing the last part of the text.

Using Layout_Streaming also requires setting the Layout_Streaming_State field.

CUNBDPRM_OutFlags - set by service

| Bit position | Name |
|---------------------|-----------------------------|
| 1xxx xxxx xxxx xxxx | CUNBDPRM_ActiveDirectional |
| x1xx xxxx xxxx xxxx | CUNBDPRM_ActiveShapeEditing |

CUNBDPRM_ActiveDirectional - set by service

Specifies if the bidi transformation included knowledge of directional code elements and proper rendering of text implies reordering of directional code elements.

- **0**: The bidi transformation does not include knowledge of directional elements.
- **1**: The bidi transformation includes knowledge of directional elements.

CUNBDPRM_ActiveShapeEditing - set by service

Specifies if the bidi transformation included knowledge of context-dependent code elements that require shaping for presentation to the target CCSID. If so, the caller must perform some shaping transformation prior to rendering the text.

- **0**: The bidi transformation does not require shape editing.
- **1**: The bidi transformation requires shape editing.

CUNBDPRM_Orientation_Src - set by caller

CUNBDPRM_Orientation_Targ - set by caller

Specifies the global directional text orientation. Possible values are:

- **ORIENTATION_LTR**: Left-to-right horizontal rows that progress from top to bottom.
- **ORIENTATION_RTL**: Right-to-left horizontal rows that progress from top to bottom.
- **ORIENTATION_TTBRL**: Top-to-bottom vertical columns that progress from right to left.
- **ORIENTATION_TTBLR**: Top-to-bottom vertical columns that progress from left to right.
- **ORIENTATION_CONTEXTUAL**: The global orientation is set according to the direction of the first significant (strong) character.

If there are no strong characters in the text and the descriptor is set to this value, the global orientation of the text is set according to the value of the CUNBDPRM_Context. This option is meaningful only for bidirectional text.

The default is ORIENTATION_LTR.

CUNBDPRM_Context_Src - set by caller

CUNBDPRM_Context_Targ - set by caller

Specifies what orientation is used when no strong character appears in the text. This is meaningful only if the corresponding CUNBDPRM_Orientation parameter is set to ORIENTATION_CONTEXTUAL. Possible values are:

- **CONTEXT_LTR**: In the absence of characters with strong directionality in the text, orientation is assumed to be left-to-right rows progressing from top to bottom.
- **CONTEXT_RTL**: In the absence of characters with strong directionality in the text, orientation is assumed to be right-to-left rows progressing from top to bottom.

The default is `CONTEXT_LTR`.

CUNBDPRM_TypeOfText_Src - set by caller

CUNBDPRM_TypeOfText_Targ - set by caller

Specifies the ordering of the directional text. Characters may have a natural orientation attached to them as described by `CUNBDPRM_Orientation`. Possible values are:

- **TEXT_VISUAL**: Code elements are stored in visually ordered segments, which can be rendered without any segment inversion. Practically the whole text could be seen as if there were no sub segments.
- **TEXT_IMPLICIT**: Code elements are stored in logically ordered segments. Logically ordered means that the order in which the characters are stored is the same as the order in which the characters are pronounced when reading the presented text or the order in which characters would be entered from a keyboard. Logical order (or logical sequence) of characters is necessary for processing purposes; for example, when there is a need to sort or index the data. Segments of reversed orientation are recognized and inverted by a content-sensitive algorithm based on the natural orientation of characters. Because there are several possible algorithms for implicit reordering of directional segments, the `ImplicitAlg` value is used when `TypeOfText` is set to `TEXT_IMPLICIT`, to indicate the actual algorithm used.
- **TEXT_EXPLICIT**: Code elements are stored in logically ordered segments with a set of embedded controls. The explicit algorithm eliminates the ambiguities that might exist in some situations when using an implicit algorithm, but it introduces the need for additional control characters in the data stream. The set of embedded controls for `TEXT_EXPLICIT` is implementation defined.

The default (for the C locale) is `TEXT_IMPLICIT`.

CUNBDPRM_ImplicitAlg_Src - set by caller

CUNBDPRM_ImplicitAlg_Targ - set by caller

Specifies the type of bidirectional implicit algorithm used in reordering and shaping of directional or context-dependent text. Possible values are:

- **ALGOR_IMPLICIT**: Directional code elements will be reordered using an implementation-defined implicit directional algorithm when converting to or from an implicit form.

Although the basic algorithm used when `ImplicitAlg` is set to `ALGOR_BASIC`, is an implicit algorithm, the fact that it recognizes some control characters, allows it to be used even when the `TypeOfText` descriptor is set to `TEXT_EXPLICIT`.

Note: When `TEXT_EXPLICIT` is used in conjunction with `ALGOR_BASIC`, the controls may temporarily change the values of swapping, numerals and `TextShaping`. The `ALGOR_IMPLICIT` value may be equal to `ALGOR_BASIC` for a given implementation. Except in this case, it is not meaningful to have `TypeOfText=TEXT_EXPLICIT` at the same time as `ImplicitAlg=ALGOR_IMPLICIT`

- **ALGOR_BASIC**: The basic algorithm is used.

The default (for the C locale) is `ALGOR_IMPLICIT`.

CUNBDPRM_Swapping_Src - set by caller

CUNBDPRM_Swapping_Targ - set by caller

Specifies whether symmetric swapping is applied to the text. A list of symmetric swapping characters is given in the ISO/IEC 10646 standard. Possible values are:

- **SWAPPING_YES**: The text conforms to symmetric swapping.
- **SWAPPING_NO**: The text does not conform to symmetric swapping.

The default (for the C locale) is `SWAPPING_NO`.

CUNBDPRM_Numerals_Src - set by caller

CUNBDPRM_Numerals_Targ - set by caller

Specifies the shaping of numerals. Possible values are:

- **NUMERALS_NOMINAL**: Nominal shaping of numerals using the portable character set (Arabic numerals).
- **NUMERALS_NATIONAL**: National shaping of numerals based on the script of the C locale.
- **NUMERALS_CONTEXTUAL**: Contextual shaping of numerals depending on the context (script) of surrounding text (such as Hindi numbers in Arabic text and Arabic numbers otherwise).

The default (for the C locale) is NUMERALS_NOMINAL.

CUNBDPRM_TextShaping_Src - set by caller

CUNBDPRM_TextShaping_Targ - set by caller

Specifies the shaping; that is, choosing (or composing) the correct shape of the text. Possible values are:

- **TEXT_SHAPED**: The text has presentation form shapes.
- **TEXT_NOMINAL**: The text is in basic form.
- **TEXT_SHFORM1**: The text is in shape form 1.
- **TEXT_SHFORM2**: The text is in shape form 2.
- **TEXT_SHFORM3**: The text is in shape form 3.
- **TEXT_SHFORM4**: The text is in shape form 4.

The set of shaping characters is limited to the CUNBCPRM_Targ_CCSID specified.

The default (for the C locale) is TEXT_SHAPED.

The term 'shape form *n*' is used to mean:

- **Arabic Script**
- **Shape form 1**: Initial form.
- **Shape form 2**: Middle form.
- **Shape form 3**: Final form.
- **Shape form 4**: Isolated form.

CUNBDPRM_ShapeCharsetSize - set by service

Specifies the size, in bytes, of the encoding of characters in the CUNBCPRM_Targ_CCSID.

CUNBDPRM_ShapeContextSize_Front - set by service

CUNBDPRM_ShapeContextSize_Back - set by service

Specifies the size of the context, in number of code elements, that must be accounted for when performing active shape editing.

CUNBDPRM_CheckMode - set by caller

Indicates the level of checking of the elements in the source buffer for shaping and reordering purposes. It also defines the behavior of the implicit algorithm with respect to standalone neutral characters (until stabilized by a new strong character). Possible values are:

- **MODE_STREAM**: The string in the source buffer is expected to have valid combinations of characters or character elements. No validation is needed before shaping or combined character cell determination. The only thing validated before the transformation is the current state of the layout object based on previous input data.

The reordering of bidirectional text will assign the nesting level of an unstabilized neutral character such that it follows the level of the previous strong character.

It is guaranteed that each shape associated with a composite sequence will occupy a single display cell.

- **MODE_EDIT**: The shaping of input text may vary depending on locale-specific validation or assumptions.

The reordering of bidirectional text will assign the nesting level of an unstabilized neutral character such that it follows the level of the global orientation.

Not all code elements of a composite sequence may be assumed to occupy a single display cell.

The default (for the C locale) is `MODE_STREAM`.

CUNBDPRM_ArabicOneCellShaping_Src - set by caller

CUNBDPRM_ArabicOneCellShaping_Targ - set by caller

Specifies which Arabic one-cell shaping transformations are performed. One-cell shaping refers to the final forms of the seen family.

The effect of this parameter depends on the setting of the `TypeOfText` parameter. Combinations are:

- **ArabicOneCellShaping_Src is `TWOCELL_SEEN`, and ArabicOneCellShaping_Targ is `ONECELL_SEEN`, and TypeOfText_Src is `TEXT_VISUAL`, and TypeOfText_Targ is `logical`:** Transformation from visual to logical converts final forms of the seen family represented by two characters (the three quarters shape and the tail character) to corresponding nominal code points represented by one character and a space replacing the tail. This space is positioned next to the seen character.
- **ArabicOneCellShaping_Src is `ONECELL_SEEN`, and ArabicOneCellShaping_Targ is `TWOCELL_SEEN`, and TypeOfText_Src is `logical`, and TypeOfText_Targ is `TEXT_VISUAL`:** In transformation from logical to visual, each character in the seen family which is to receive a final form is converted to the corresponding final form of the seen family that is represented by two characters, consuming an existing space next to the seen character. If there is no space available, it will be converted to one character only which is the three quarters shape seen.
- **Other settings:** Seen tail characters remain as is.

CUNBDPRM_WordBreak_Src - set by caller

CUNBDPRM_WordBreak_Targ - set by caller

Specifies if the service is to transform each word in isolation from adjacent words based on whitespace delimiters.

Combinations are:

- **WordBreak_Src is `NO_BREAK`, and WordBreak_Targ is `BREAK`:** Transform each word in isolation from adjacent words based on whitespace delimiters.
- **Other settings:** Do not transform each word in isolation from adjacent words based on whitespace delimiters.

CUNBDPRM_LamAlefEditMode_Src - set by caller

CUNBDPRM_LamAlefEditMode_Targ - set by caller

Specifies which Lam-Alef edit mode transformations are performed.

Combinations are:

- **LamAlefEditMode_Src is `LamAlefOff`, and LamAlefEditMode_Targ is `LamAlefOff`:**
 - When transforming from visual to logical layouts, Lam-Alef characters are expanded to Lam plus Alef consuming an existing blank space next to it. If no blank space is available, the Lam-Alef character remains as is.
 - When transforming from logical to visual layouts, Lam plus Alef sequences are compressed to a unique Lam-Alef character; the space resulting from the Lam-Alef compression is positioned next to each generated Lam-Alef character.
- **LamAlefEditMode_Src is `LamAlefOff`, and LamAlefEditMode_Targ is `LamAlefOn`:**
 - When transforming from visual to implicit layouts, Lam-Alef characters are expanded to Lam plus Alef consuming a blank space at the absolute end of the buffer. If no blank space is available, the Lam-Alef character remains as is.
 - When transforming from implicit to visual layouts, Lam plus Alef sequences are compressed to a unique Lam-Alef character; the space resulting from Lam-Alef compression is positioned at the absolute end of the buffer.

- **LamAlefEditMode_Src is LamAlefOff, and LamAlefEditMode_Targ is LamAlefAuto:** For each LAMALEF character found, expand LAMALEF using space at end. If there is no space at end, use spaces at beginning of the buffer. If there is no space at the beginning of the buffer, use spaces at the near (for example, the space after the LAMALEF character).
- **Other settings:** Lam Alef characters remain as is.

CUNBDPRM_YehHamzaMode_Src - set by caller

CUNBDPRM_YehHamzaMode_Targ - set by caller

Specifies which YehHamza edit mode transformations are performed. Possible values are:

- **ONECELL_YAHHAMZA:** The Yeh-Hamza final form is represented as one character.
- **TWOCELL_YAHHAMZA:** The Yeh-Hamza final form is represented as two characters.

The default value for CUNBDPRM_YehHamzaMode is TWOCELL_YAHHAMZA, if the CCSID is 00420 or 00864. Otherwise, it is ONECELL_YAHHAMZA.

CUNBDPRM_TailEditMode_Src - set by caller

CUNBDPRM_TailEditMode_Targ - set by caller

Specifies which Tail edit mode transformations are performed. Possible values are:

- **NEW_TAIL:** A newly defined Tail character (U+FE73) in Unicode 3.2 to replace the legacy Seen family Tail character.
- **OLD_TAIL:** A legacy Seen family tail character (U+200B).

The default value for CUNBDPRM_TailEditMode is OLD_TAIL.

CUNBDPRM_TashkeelEditMode_Src - set by caller

CUNBDPRM_TashkeelEditMode_Targ - set by caller

Specifies which Tashkeel edit mode transformations are performed. Possible values are:

- **TASHKEELBEGIN:** All Tashkeel characters (except for Shadda) are replaced by spaces. The resulting spaces are moved to the beginning of the buffer.
- **TASHKEELEND:** All Tashkeel characters (except for Shadda) are replaced by spaces. The resulting spaces are moved to the end of the buffer.
- **TASHKEELREPLACEWITHTATWEEL:** All Tashkeel characters (except for Shadda) are ignored and reseize the data buffer. This is done only when the output codepage is 420 or 864.
- **TASHKEELRESIZE:** All Tashkeel characters (except for Shadda) are ignored and reseize the data buffer. This is done only when the output codepage is 420 or 864.
- **TASHKEELISOLATED:** All Tashkeel or Tatweel characters (except for Shadda) are ignored and reseize the data buffer.

The default value for CUNBDPRM_TashkeelEditMode is TASHKEELEND.

CUNBDPRM_InpToOut_Ptr - set by caller

Specifies a buffer to receive a cross-reference from each Src_Buf code element to the transformed data. The cross-reference relates to the data in Src_Buf starting with the first element that InpBufIndex points to (and not necessarily starting from the beginning of the Src_Buf).

If not a NULL pointer, it points to an array of values with the same number of bytes in Src_Buf starting with the one pointed by InpBufIndex and up to the end of the substring in the buffer. On output, the *n*th value in InpToOut corresponds to the *n*th byte in Src_Buf. This value is the index (in units of bytes) in Targ_Buf that identifies the transformed element of the *n*th byte in Src_Buf. In the case of multibyte encoding, the index points (for each of the bytes of a code element in the Src_Buf) to the first byte of the transformed code element in the Targ_Buf.

InpToOut may be specified as NULL if no index array from Src_Buf to Targ_Buf is desired.

CUNBDPRM_OutToInp_Ptr - set by caller

Specifies a buffer to receive a cross-reference from each Targ_Buf code element to the source buffer. The cross-reference relates to the data in Src_Buf starting with the first element that InpBufIndex points to (and not necessarily starting from the beginning of the Src_Buf).

If not a NULL pointer, it points to an array of values with the same number of bytes in Targ_Buf. On output, the *n*th value in OutToInp corresponds to the *n*th byte in Targ_Buf. This value is the index (in units of bytes) in Src_Buf that identifies the source of the transformed element of the *n*th byte in Targ_Buf. In the case of multibyte encoding, the index points (for each of the bytes of a code element in the Targ_Buf) to the first byte of the source of the transformed code element in the Src_Buf.

OutToImp may be specified as NULL if no index array from Targ_Buf to Src_Buf is desired.

CUNBDPRM_BidiLvl_Ptr - set by caller

A weighted value that represents peculiar input string transformation properties with different connotations as explained below.

If this argument is not a NULL pointer, it represents an array of values with the same number of elements as the Src_Buf before the transformation. Each byte will contain relevant BidiLvl information of the corresponding element in Src_Buf starting from the element pointed by InpBufIndex. The four rightmost bits of each BidiLvl byte will contain information for bidirectional environments (when ActiveDirectional is true) and they will mean NestingLevels. The possible value from 0 to 15 represents the nesting level of the corresponding element in the Src_Buf starting from the element pointed by InpBufIndex. If ActiveDirectional is false, the content of NestingLevel bits will be ignored. The leftmost bit of each BidiLvl byte will contain a new cell indicator for composed character environments and will have a value of either 1 (for an element in Src_Buf that is transformed to the beginning of a new cell) or zero (for the zero-length composing character elements, when these are grouped into the same presentation cell with a non-composing character). Each element of BidiLvl pertains to the elements in the Src_Buf starting from the element pointed by InpBufIndex. Remember that this is not necessarily the beginning of SrcBuf.

If none of the transformation properties is required, the argument property can be NULL.

The use of BidiLvl can be enhanced in the future to pertain to other possible usage in other environments.

CUNBDPRM_InpBufIndex - set by caller, updated by service

InpBufIndex is an offset value to the location of the transformed text. When the bidi service is invoked, InpBufIndex contains the offset to the element in Src_Buf that will be transformed first. Note: This is not necessarily the first element in Src_Buf. At the return from the transformation, InpBufIndex contains the offset to the first element in the Src_Buf that has not been transformed. If the entire substring has been transformed successfully, InpBufIndex will be incremented by the amount defined by Src_Buf_Len.

Set by caller. The service updates the offset value.

CUNBDPRM_Streaming_Processed_Length - set by service

Specifies the amount of source text, in bytes, that layout streaming processed. Set by service when Layout_Streaming is set.

CUNBDPRM_Layout_Streaming_State - set by caller, updated by service

Contains the state of the bidi transformation between calls to the service when Layout_Streaming is used.

The caller should set this area to all zero bytes the first time calling the service with Layout_Streaming and then not modify the value for subsequent calls to the service that use the same layout streaming operation. When using layout streaming, the last call in the sequence is with the Layout_Streaming bit turned off. The caller should not modify the content of the Layout_Streaming_State until after that call returns.

Set by caller and updated by the service when Layout_Streaming is used. Ignored when Layout_Streaming is not used.

CUNBDPRM_Bidi_Keyword - set by caller

This is a short form for extended bidi settings.

Note: Short path settings have higher priority over defaults and long path settings.

Format of CUNBDPRM_Bidi_Keyword:

```
Key1+Value_Key2+Value_Key3+Value...
```

Note:

1. Since most attributes (except for LayoutOptions and CheckMode attributes) can apply to both the source and target data, the second letter in the *key* indicates whether the attributes is for the source (S) or target (T) buffer.
2. If the same key is specified more than once, the last specified value is used.

In the example:

```
0S0_0T1_TS1_TT2
```

- Orientation of the source buffer is LTR.
- Orientation of the target buffer is RTL.
- Type of text of the source buffer is implicit.
- Type of text of the target buffer is explicit.

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|----------------|--|---------------------------------|---|
| LayoutOptions | Lx | 0-252 | <p>Layout options. Values:</p> <ul style="list-style-type: none"> • 1... (128) = CUNBDPRM_Layout_Roundtrip • .1.. (64) = CUNBDPRM_Layout_WinCompat • ..1. (32) = CUNBDPRM_Layout_ImpToImp • ...1 (16) = CUNBDPRM_Layout_Remove_Marks • 1... (8) = CUNBDPRM_Layout_Insert_Marks •1.. (4) = CUNBDPRM_Layout_Streaming <p>Example of Roundtrip and ImpToImp (or Logical to Logical):</p> <pre>L160</pre> <p>For long path equivalent setting, see CUNBDPRM_Layout_Options description.</p> |

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|----------------|--|---------------------------------|---|
| Orientation | Obx | 0-4 | <p>The direction of the text. Values:</p> <ul style="list-style-type: none"> • 0 = ORIENTATION_LTR (Input/Output Default) • 1 = ORIENTATION_RTL • 2 = ORIENTATION_TTBLR • 3 = ORIENTATION_TTBRL • 4 = ORIENTATION_CONTEXTUAL <p>The mappings between short form and long form are defined by BIDI_ORIENTATION in the interface definition file CUNBCIDF.</p> |
| Context | Cbx | 0-1 | <p>Contextual orientation when the orientation attribute is set to ORIENTATION_CONTEXTUAL. Values:</p> <ul style="list-style-type: none"> • 0 = CONTEXT_LTR (Input/Output Default) • 1 = CONTEXT_RTL <p>The mappings between short form and long form are defined by BIDI_CONTEXT in the interface definition file CUNBCIDF.</p> |
| TypeofText | Tbx | 0-2 | <p>Type of the text. Values:</p> <ul style="list-style-type: none"> • 0 = TEXT_VISUAL (Output default) • 1 = TEXT_IMPLICIT (Input default) • 2 = TEXT_EXPLICIT <p>The mappings between short form and long form are defined by BIDI_TEXT_TYPE in the interface definition file CUNBCIDF.</p> |
| ImplicitAlg | Ibx | 0-1 | <p>Implicit algorithm used in the source/target buffer. Values:</p> <ul style="list-style-type: none"> • 0 = ALGOR_BASIC (Input/Output Default) • 1 = ALGOR_IMPLICIT <p>The mappings between short form and long form are defined by BIDI_IMPALG in the interface definition file CUNBCIDF.</p> |

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|----------------|--|---------------------------------|--|
| Swapping | Sbx | 0-1 | <p>Specifies whether symmetric swapping is enabled. Values:</p> <ul style="list-style-type: none"> • 0 = SWAPPING_NO (Output default) • 1 = SWAPPING_YES (Input default) <p>The mappings between short form and long form are defined by BIDI_SWAPPING in the interface definition file CUNBCIDF.</p> |
| Numerals | Nbx | 0-3 | <p>How numerals are shaped. Values:</p> <ul style="list-style-type: none"> • 0 = NUMERALS_NOMINAL (Input default. Output default in Hebrew locale.) • 1 = NUMERALS_NATIONAL • 2 = NUMERALS_CONTEXTUAL (Output default in Arabic locale) • 3 = NUMERALS_NONE <p>The mappings between short form and long form are defined by BIDI_NUMERALS in the interface definition file CUNBCIDF.</p> |
| TextShaping | Ebx | 0-7 | <p>Specifies whether text to be shaped. Values:</p> <ul style="list-style-type: none"> • 0 = TEXT_SHAPED (Output default in Arabic locale) • 1 = TEXT_NOMINAL (Input default, Output default in Hebrew locale) • 2 = TEXT_SHFORM1 • 3 = TEXT_SHFORM2 • 4 = TEXT_SHFORM3 • 5 = TEXT_SHFORM4 • 6 = TEXT_STANDARD • 7 = TEXT_COMPOSED <p>The mappings between short form and long form are defined by BIDI_SHAPING in the interface definition file CUNBCIDF.</p> |

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|-----------------------|--|--|---|
| CheckMode | Hx | 0-1 | <p>Level of Bidi checking (apply to both source and target). Values:</p> <ul style="list-style-type: none"> • 0 = MODE_STREAM • 1 = MODE_EDIT (Input/Output default) <p>The mappings between short form and long form are defined by BIDI_CHECKMODE in the interface definition file CUNBCIDF.</p> |
| WordBreak | Wbx | 0-1 | <p>Word break. Values:</p> <ul style="list-style-type: none"> • 0 = WORD_BREAK • 1 = NO_BREAK (Input/Output default) <p>The mappings between short form and long form are defined by BIDI_WORDBREAK in the interface definition file CUNBCIDF.</p> |
| LamAlefEdit | Fbx | 0-5 | <p>LamAlef edit mode. Values:</p> <ul style="list-style-type: none"> • 0 = LamAlefOn • 1 = LamAlefBegin • 2 = LamAlefResize • 3 = LamAlefNear • 4 = LamAlefAuto (Input/Output default) • 5 = LamAlefOff <p>The mappings between short form and long form are defined by BIDI_LAMALEF in the interface definition file CUNBCIDF.</p> |
| ArabicOneCell | Abx | 0-1 | <p>Arabic one-cell shaping. Values:</p> <ul style="list-style-type: none"> • 0 = ONECELL_SEEN (Input default. Output default for Hebrew locale.) • 1 = TWOCELL_SEEN (Output default for Arabic locale.) <p>The mappings between short form and long form are defined by BIDI_ONECELL in the interface definition file CUNBCIDF.</p> |

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|----------------|--|---------------------------------|--|
| TailMode | Mbx | 0-1 | <p>Tail edit mode. Values:</p> <ul style="list-style-type: none"> • 0 = NEW_TAIL • 1 = OLD_TAIL <p>The mappings between short form and long form are defined by BIDI_TAIL in the interface definition file CUNBCIDF.</p> |
| TashkeelMode | Kbx | 0-4 | <p>Tashkeel edit mode. Values:</p> <ul style="list-style-type: none"> • 0 = TashkeelBegin • 1 = TashkeelEnd • 2 = TashkeelReplaceWithTatweel • 3 = TashkeelResize • 4 = TashkeelIsolated <p>The mappings between short form and long form are defined by BIDI_TASHKEEL in the interface definition file CUNBCIDF.</p> |
| YehHamza | Ybx | 0-1 | <p>YehHamza edit mode. Values:</p> <ul style="list-style-type: none"> • 0 = ONECELL_YEHAMZA (Input default. Output default for Hebrew locale.) • 1 = TWOCELL_YEHAMZA (Output default for Arabic locale.) <p>The mappings between short form and long form are defined by BIDI_YEHAMZA in the interface definition file CUNBCIDF.</p> |

AMODE(64)

| Table 5. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(64) | | | | | | |
|---|------------|-----------|-----------------|----------|------------------|--|
| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
| 0 | (0) | STRUCTURE | | DWORD | CUN4BDPR | Extended bidi parameter area for 31-bit character conversion |
| | (0) | UNSIGNED | 4 | | CUN4BDPR_Version | Version of the parameter area |
| | (4) | UNSIGNED | 4 | | CUN4BDPR_Len | Length, in bytes, of the parameter area |
| | (8) | BITSTRING | 1 | | CUN4BDPR_InFlags | Input flags |

Table 5. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(64) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|--------------------|-----------------|----------|------------------------------|---|
| | | 1... .. | 1 | | CUN4BDPR_XOpen_Defaults | Specifies X/Open portable layout option defaults |
| | | .1.. .. | 1 | | CUN4BDPR_KBS_Defaults | Specifies Unicode Services knowledge base defaults |
| | | ..1. | 1 | | CUN4BDPR_Keyword | Specifies bidi keyword |
| | | ...1 | 1 | | CUN4BDPR_From_wtransform | Reserved for Unicode Services use. This should not be set by users. |
| | (9) | BITSTRING | 1 | | CUN4BDPR_Layout_Options | Layout options |
| | | 1... .. | 1 | | CUN4BDPR_Layout_Roundtrip | Specifies if round trip processing is to be used |
| | | .1.. .. | 1 | | CUN4BDPR_Layout_WinCompat | Specifies if the WinCompat mode is to be used |
| | | ..1. | 1 | | CUN4BDPR_Layout_ImpToImp | Specifies if a 'logical to logical' transformation is to be performed |
| | | ...1 | 1 | | CUN4BDPR_Layout_Remove_Marks | Specifies if all bidi marks will be removed |
| | | 1... | 1 | | CUN4BDPR_Layout_Insert_Marks | Specifies if bidi marks are to be inserted |
| | |1.. | 1 | | CUN4BDPR_Layout_Streaming | Specifies if layout streaming is to be used |
| | (A) | BITSTRING | 2 | | CUN4BDPR_OutFlags | Output flags |
| | | 1... | | | CUN4BDPR_ActiveDirectional | Specifies if directional elements were used |
| | | .1.. | | | CUN4BDPR_ActiveShapeEditing | Specifies if caller must perform shape editing |
| | (C) | CHARACTER | 4 | | Reserved | |
| | (10) | UNSIGNED | 4 | | CUN4BDPR_Orientation_Src | Orientation of the source buffer |
| | (14) | UNSIGNED | 4 | | CUN4BDPR_Orientation_Targ | Orientation of the target buffer |
| | (18) | UNSIGNED | 4 | | CUN4BDPR_Context_Src | Context of the source buffer |
| | (1C) | UNSIGNED | 4 | | CUN4BDPR_Context_Targ | Context of the target buffer |
| | (20) | UNSIGNED | 4 | | CUN4BDPR_TypeOfText_Src | Type of text of the source buffer |
| | (24) | UNSIGNED | 4 | | CUN4BDPR_TypeOfText_Targ | Type of text of the target buffer |
| | (28) | UNSIGNED | 4 | | CUN4BDPR_ImplicitAlg_Src | Implicit algorithm used in the source buffer |
| | (2C) | UNSIGNED | 4 | | CUN4BDPR_ImplicitAlg_Targ | Implicit algorithm used in the target buffer |
| | (30) | UNSIGNED | 4 | | CUN4BDPR_Swapping_Src | Swapping used in the source buffer |
| | (34) | UNSIGNED | 4 | | CUN4BDPR_Swapping_Targ | Swapping used in the target buffer |

Table 5. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(64) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|----------|-----------------|----------|-------------------------------------|--|
| | (38) | UNSIGNED | 4 | | CUN4BDPR_Numerals_Src | Numerals used in the source buffer |
| | (3C) | UNSIGNED | 4 | | CUN4BDPR_Numerals_Targ | Numerals used in the target buffer |
| | (40) | UNSIGNED | 4 | | CUN4BDPR_TextShaping_Src | Text shaping used in the source buffer |
| | (44) | UNSIGNED | 4 | | CUN4BDPR_TextShaping_Targ | Text shaping used in the target buffer |
| | (48) | UNSIGNED | 4 | | CUN4BDPR_ShapeCharsetSize | Size of elements of the character set |
| | (4C) | UNSIGNED | 4 | | CUN4BDPR_ShapeContextSize_Front | Number of code elements required for shape editing |
| | (50) | UNSIGNED | 4 | | CUN4BDPR_ShapeContextSize_Back | Number of code elements required for shape editing |
| | (54) | UNSIGNED | 4 | | CUN4BDPR_CheckMode | Level of bidi checking |
| | (58) | UNSIGNED | 8 | | CUN4BDPR_InpBufIndex | Bidi input buffer index |
| | (60) | UNSIGNED | 8 | | CUN4BDPR_Streaming_Processed_Length | Bidi streaming processed length |
| | (68) | UNSIGNED | 4 | | CUN4BDPR_ArabicOneCellShaping_Src | Arabic one-cell shaping used in the source buffer |
| | (6C) | UNSIGNED | 4 | | CUN4BDPR_ArabicOneCellShaping_Targ | Arabic one-cell shaping used in the target buffer |
| | (70) | UNSIGNED | 4 | | CUN4BDPR_WordBreak_Src | Word break used in the source buffer |
| | (74) | UNSIGNED | 4 | | CUN4BDPR_WordBreak_Targ | Word break used in the target buffer |
| | (78) | UNSIGNED | 4 | | CUN4BDPR_LamAlefEditMode_Src | LamAlef edit mode used in the source buffer |
| | (7C) | UNSIGNED | 4 | | CUN4BDPR_LamAlefEditMode_Targ | LamAlef edit mode used in the target buffer |
| | (80) | UNSIGNED | 4 | | CUN4BDPR_YehHamzaMode_Src | YehHamza edit mode used in the source buffer |
| | (84) | UNSIGNED | 4 | | CUN4BDPR_YehHamzaMode_Targ | YehHamza edit mode used in the target buffer |
| | (88) | UNSIGNED | 4 | | CUN4BDPR_TailEditMode_Src | Tail edit mode used in the source buffer |
| | (8C) | UNSIGNED | 4 | | CUN4BDPR_TailEditMode_Targ | Tail edit mode used in the target buffer |
| | (90) | UNSIGNED | 4 | | CUN4BDPR_TashkeelEditMode_Src | Tashkeel edit mode used in the source buffer |
| | (94) | UNSIGNED | 4 | | CUN4BDPR_TashkeelEditMode_Targ | Tashkeel edit mode used in the target buffer |
| | (98) | ADDRESS | 8 | | CUN4BDPR_InpToOut_Ptr | Bidi input to output buffer pointer |
| | (A0) | ADDRESS | 8 | | CUN4BDPR_OutToInp_Ptr | Bidi output to input buffer pointer |

Table 5. Mapping of parameters in HLASM for the extended bidi parameter area of character conversion in AMODE(64) (continued)

| Offset Dec | Offset Hex | Type | Length in bytes | Boundary | Name | Description |
|------------|------------|-----------|-----------------|----------|---------------------------------|---|
| | (A8) | ADDRESS | 8 | | CUN4BDPR_BidiLvl_Ptr | BidiLvl property pointer |
| | (B0) | CHARACTER | 64 | | CUN4BDPR_Layout_Streaming_State | State of the layout streaming operation |
| | (F0) | CHARACTER | 128 | | CUN4BDPR_Bidi_Keyword | Short form keyword |
| | (170) | CHARACTER | 64 | | * | Reserved |
| | (1B0) | | 0 | | CUN4BDPR_End | End of CUN4BDPR |

Description of parameters in area CUN4BDPR

This description applies to HLASM.

CUN4BDPR_Version - set by caller

Specifies the version of the parameter area. Use version 1.

CUN4BDPR_Length - set by caller

Specifies the length of the parameter area, in bytes. Use constant CUN4BDPR_Len.

CUN4BDPR_InFlags - set by caller (except for CUN4BDPR_From_wtransform)

| Bit position | Name |
|--------------|--------------------------|
| 1xxx xxxx | CUN4BDPR_XOpen_Defaults |
| x1xx xxxx | CUN4BDPR_KBS_Defaults |
| xx1x xxxx | CUN4BDPR_Keyword |
| xxx1 xxxx | CUN4BDPR_From_wtransform |

CUN4BDPR_XOpen_Defaults - set by caller

Specifies whether or not to use default settings for the X/Open portable layout options. Possible values are:

- **0**: Do not use default settings for the X/Open portable layout options.
- **1**: Use default settings for the X/Open portable layout options.

Note: The settings defined in the short-form keyword CUN4BDPR_Bidi_Keyword have higher priority over the defaults. The attributes specified in the bidi keyword will overlay the default attributes.

CUN4BDPR_KBS_Defaults - set by caller

Specifies whether or not to use default settings from the Unicode Services knowledge base to set the X/Open portable layout options. Possible values are:

- **0**: Do not use default settings from the Unicode Services knowledge base to set the X/Open portable layout options.
- **1**: Use default settings from the Unicode Services knowledge base to set the X/Open portable layout options.

Note: This flag is ignored if CUN4BDPR_XOpen_Defaults is ON. If CUN4BDPR_XOpen_Defaults is OFF and CUN4BDPR_KBS_Defaults is ON, the defaults defined in the Unicode Services knowledge base will be used. The bidi string types and associated attributes defined in the knowledge

base are based on the input or output CCSID. The settings defined in the short-form keyword CUN4BDPR_Bidi_Keyword have higher priority over the default attributes.

CUN4BDPR_Keyword - set by caller

Specifies whether or not to use the short form keyword to set the X/Open portable layout options. Possible values are:

- **0**: Do not use the short form keyword to set the X/Open portable layout options.
- **1**: Use the short form keyword to set the X/Open portable layout options.

Note: This flag must be set to ON when the CUN4BDPR_Bidi_Keyword is used.

CUN4BDPR_From_wtransform - set by service

This flag is reserved for internal Unicode Services use. It should not be set by the caller.

CUN4BDPR_Layout_Options - set by caller

| Bit position | Name |
|--------------|------------------------------|
| 1xxx xxxx | CUN4BDPR_Layout_Roundtrip |
| x1xx xxxx | CUN4BDPR_Layout_WinCompat |
| xx1x xxxx | CUN4BDPR_Layout_ImpToImp |
| xxx1 xxxx | CUN4BDPR_Layout_Remove_Marks |
| xxxx 1xxx | CUN4BDPR_Layout_Insert_Marks |
| xxxx x1xx | CUN4BDPR_Layout_Streaming |

CUN4BDPR_Layout_Roundtrip - set by caller

Specifies if numbers located between LTR text and RTL text are associated with the RTL text. This makes the algorithm reversible and makes it useful when round trip (from visual to logical and back to visual) must be achieved without adding LRM characters. However, this is a variation from the standard Unicode bidi algorithm. Possible values are:

- **0**: Numbers are not associated with the RTL text.
- **1**: Numbers are associated with the RTL text.

CUN4BDPR_Layout_WinCompat - set by caller

Specifies if the algorithm used to perform bidi transformations should approximate the algorithm used in Microsoft Windows XP, rather than strictly conform to the Unicode bidi algorithm. Possible values are:

- **0**: Do not approximate the Microsoft algorithm.
- **1**: Approximate the Microsoft algorithm.

CUN4BDPR_Layout_ImpToImp - set by caller

Specifies if a logical to logical transformation is to be performed:

- If the source orientation is LTR, the source text will be handled as LTR logical text and will be transformed to the RTL logical text which has the same LTR visual display.
- If the source orientation is RTL, the source text will be handled as RTL logical text and will be transformed to the LTR logical text which has the same LTR visual display.

This mode may be needed when logical text, which is basically Arabic or Hebrew, with possible included numbers or phrases in English, has to be displayed as if it had LTR orientation. This can happen if the displaying application treats all text as if it was basically LTR. This mode may also be needed in the reverse case, when logical text which is basically English, with possible included phrases in Arabic or Hebrew, has to be displayed as if it had RTL orientation. The problem may be handled by transforming the source text with this option before displaying it, so that it will be displayed properly. Possible values are:

- **0**: Logical to logical transformation is not to be performed.
- **1**: Logical to logical transformation is to be performed.

CUN4BDPR_Layout_Remove_Marks - set by caller

Specifies if all bidi marks (LRM or RLM) will be removed from the output text when performing a transformation. Possible values are:

- **0**: Bidi marks are not to be removed from the output text.
- **1**: Bidi marks are to be removed from the output text. The corresponding entries in the InpToOut map are set equal to the maximum value. This option should not be specified together with option `Layout_Insert_Marks`, and it overrides it.

CUN4BDPR_Layout_Insert_Marks - set by caller

Specifies if bidi marks (LRM or RLM) are to be inserted when needed to ensure correct results when reordering to an implicit order. This option is meaningful only when performing a transformation from visually ordered to implicitly ordered text. Possible values are:

- **0**: Do not insert bidi marks.
- **1**: Insert bidi marks. A minimum number of LRM or RLM characters will be added to the source text after reordering it so as to ensure round trip. For example, when applying the inverse transformation on the resulting implicit text with removal of bidi marks (option `Layout_Remove_Marks`), the result will be identical to the source text in the first transformation. The LRM and RLM characters, which are added in the output text, have no matching character in the source text. The corresponding entries in the OutToInp map are set equal to the maximum value.

Set by caller. Ignored if specified together with `CUN4BDPR_Layout_Remove_Marks`.

CUN4BDPR_Layout_Streaming - set by caller

Specifies if the caller is interested in using layout streaming. Layout streaming processes large text objects into parts using the piece by piece technique. The results of the successive calls are expected to be concatenated by the caller. Only the call for the last part will have this option bit off. Possible values are:

- **0**: Do not use layout streaming.
- **1**: Attempt to use layout streaming. The transform operation may process less than the full source text in order to truncate the text at a meaningful boundary. The caller must read the value in `CUN4BDPR_Streaming_Processed_Length` immediately after performing the transform in order to determine how much of the source text has been processed. Source text beyond that length should be resubmitted in following transform operations. If the last character of the source text constitutes a reasonable boundary, the whole text will be processed at once. If no where in the source text there exists such a reasonable boundary, the processed length will be zero. The caller should check for such an occurrence and do one of the following:
 - Submit a larger amount of text with a better chance to include a reasonable boundary.
 - Resubmit the same text after turning off this option.

In all cases, this option should be turned off before processing the last part of the text.

Using `Layout_Streaming` also requires setting the `Layout_Streaming_State` field.

CUN4BDPR_OutFlags - set by service

| Bit position | Name |
|---------------------|-----------------------------|
| 1xxx xxxx xxxx xxxx | CUN4BDPR_ActiveDirectional |
| x1xx xxxx xxxx xxxx | CUN4BDPR_ActiveShapeEditing |

CUN4BDPR_ActiveDirectional - set by service

Specifies if the bidi transformation included knowledge of directional code elements and proper rendering of text implies reordering of directional code elements.

- **0:** The bidi transformation does not include knowledge of directional elements.
- **1:** The bidi transformation includes knowledge of directional elements.

CUN4BDPR_ActiveShapeEditing - set by service

Specifies if the bidi transformation included knowledge of context-dependent code elements that require shaping for presentation to the target CCSID. If so, the caller must perform some shaping transformation prior to rendering the text.

- **0:** The bidi transformation does not require shape editing.
- **1:** The bidi transformation requires shape editing.

CUN4BDPR_Orientation_Src - set by caller**CUN4BDPR_Orientation_Targ - set by caller**

Specifies the global directional text orientation. Possible values are:

- **ORIENTATION_LTR:** Left-to-right horizontal rows that progress from top to bottom.
- **ORIENTATION_RTL:** Right-to-left horizontal rows that progress from top to bottom.
- **ORIENTATION_TTBRL:** Top-to-bottom vertical columns that progress from right to left.
- **ORIENTATION_TTBRLR:** Top-to-bottom vertical columns that progress from left to right.
- **ORIENTATION_CONTEXTUAL:** The global orientation is set according to the direction of the first significant (strong) character.

If there are no strong characters in the text and the descriptor is set to this value, the global orientation of the text is set according to the value of the CUN4BDPR_Context. This option is meaningful only for bidirectional text.

The default is ORIENTATION_LTR.

CUN4BDPR_Context_Src - set by caller**CUN4BDPR_Context_Targ - set by caller**

Specifies what orientation is used when no strong character appears in the text. This is meaningful only if the corresponding CUN4BDPR_Orientation parameter is set to ORIENTATION_CONTEXTUAL. Possible values are:

- **CONTEXT_LTR:** In the absence of characters with strong directionality in the text, orientation is assumed to be left-to-right rows progressing from top to bottom.
- **CONTEXT_RTL:** In the absence of characters with strong directionality in the text, orientation is assumed to be right-to-left rows progressing from top to bottom.

The default is CONTEXT_LTR.

CUN4BDPR_TypeOfText_Src - set by caller**CUN4BDPR_TypeOfText_Targ - set by caller**

Specifies the ordering of the directional text. Characters may have a natural orientation attached to them as described by CUN4BDPR_Orientation. Possible values are:

- **TEXT_VISUAL:** Code elements are stored in visually ordered segments, which can be rendered without any segment inversion. Practically the whole text could be seen as if there were no sub segments.
- **TEXT_IMPLICIT:** Code elements are stored in logically ordered segments. Logically ordered means that the order in which the characters are stored is the same as the order in which the characters are pronounced when reading the presented text or the order in which characters would be entered from a keyboard. Logical order (or logical sequence) of characters is necessary for processing purposes; for example, when there is a need to sort or index the data. Segments of reversed orientation are recognized and inverted by a content-sensitive algorithm based on the natural orientation of characters. Because there are several possible algorithms for implicit reordering of directional segments, the ImplicitAlg value is used when TypeOfText is set to TEXT_IMPLICIT, to indicate the actual algorithm used.
- **TEXT_EXPLICIT:** Code elements are stored in logically ordered segments with a set of embedded controls. The explicit algorithm eliminates the ambiguities that might exist in some situations when using an implicit algorithm, but it introduces the need for additional control characters in the data stream. The set of embedded controls for TEXT_EXPLICIT is implementation defined.

The default (for the C locale) is TEXT_IMPLICIT.

CUN4BDPR_ImplicitAlg_Src - set by caller

CUN4BDPR_ImplicitAlg_Targ - set by caller

Specifies the type of bidirectional implicit algorithm used in reordering and shaping of directional or context-dependent text. Possible values are:

- **ALGOR_IMPLICIT:** Directional code elements will be reordered using an implementation-defined implicit directional algorithm when converting to or from an implicit form.

Although the basic algorithm used when ImplicitAlg is set to ALGOR_BASIC, is an implicit algorithm, the fact that it recognizes some control characters, allows it to be used even when the TypeOfText descriptor is set to TEXT_EXPLICIT.

Note: When TEXT_EXPLICIT is used in conjunction with ALGOR_BASIC, the controls may temporarily change the values of swapping, numerals and TextShaping. The ALGOR_IMPLICIT value may be equal to ALGOR_BASIC for a given implementation. Except in this case, it is not meaningful to have TypeOfText=TEXT_EXPLICIT at the same time as ImplicitAlg=ALGOR_IMPLICIT

- **ALGOR_BASIC:** The basic algorithm is used.

The default (for the C locale) is ALGOR_IMPLICIT.

CUN4BDPR_Swapping_Src - set by caller

CUN4BDPR_Swapping_Targ - set by caller

Specifies whether symmetric swapping is applied to the text. A list of symmetric swapping characters is given in the ISO/IEC 10646 standard. Possible values are:

- **SWAPPING_YES:** The text conforms to symmetric swapping.
- **SWAPPING_NO:** The text does not conform to symmetric swapping.

The default (for the C locale) is SWAPPING_NO.

CUN4BDPR_Numerals_Src - set by caller

CUN4BDPR_Numerals_Targ - set by caller

Specifies the shaping of numerals. Possible values are:

- **NUMERALS_NOMINAL:** Nominal shaping of numerals using the portable character set (Arabic numerals).
- **NUMERALS_NATIONAL:** National shaping of numerals based on the script of the C locale.
- **NUMERALS_CONTEXTUAL:** Contextual shaping of numerals depending on the context (script) of surrounding text (such as Hindi numbers in Arabic text and Arabic numbers otherwise).

The default (for the C locale) is NUMERALS_NOMINAL.

CUN4BDPR_TextShaping_Src - set by caller**CUN4BDPR_TextShaping_Targ - set by caller**

Specifies the shaping; that is, choosing (or composing) the correct shape of the text. Possible values are:

- **TEXT_SHAPED:** The text has presentation form shapes.
- **TEXT_NOMINAL:** The text is in basic form.
- **TEXT_SHFORM1:** The text is in shape form 1.
- **TEXT_SHFORM2:** The text is in shape form 2.
- **TEXT_SHFORM3:** The text is in shape form 3.
- **TEXT_SHFORM4:** The text is in shape form 4.

The set of shaping characters is limited to the CUN4BDPR_Targ_CCSID specified.

The default (for the C locale) is TEXT_SHAPED.

The term 'shape form *n*' is used to mean:

- **Arabic Script**
- **Shape form 1:** Initial form.
- **Shape form 2:** Middle form.
- **Shape form 3:** Final form.
- **Shape form 4:** Isolated form.

CUN4BDPR_ShapeCharsetSize - set by service

Specifies the size, in bytes, of the encoding of characters in the CUN4BDPR_Targ_CCSID.

CUN4BDPR_ShapeContextSize_Front - set by service**CUN4BDPR_ShapeContextSize_Back - set by service**

Specifies the size of the context, in number of code elements, that must be accounted for when performing active shape editing.

CUN4BDPR_CheckMode - set by caller

Indicates the level of checking of the elements in the source buffer for shaping and reordering purposes. It also defines the behavior of the implicit algorithm with respect to standalone neutral characters (until stabilized by a new strong character). Possible values are:

- **MODE_STREAM:** The string in the source buffer is expected to have valid combinations of characters or character elements. No validation is needed before shaping or combined character cell determination. The only thing validated before the transformation is the current state of the layout object based on previous input data.

The reordering of bidirectional text will assign the nesting level of an unstabilized neutral character such that it follows the level of the previous strong character.

It is guaranteed that each shape associated with a composite sequence will occupy a single display cell.

- **MODE_EDIT:** The shaping of input text may vary depending on locale-specific validation or assumptions.

The reordering of bidirectional text will assign the nesting level of an unstabilized neutral character such that it follows the level of the global orientation.

Not all code elements of a composite sequence may be assumed to occupy a single display cell.

The default (for the C locale) is MODE_STREAM.

CUN4BDPR_ArabicOneCellShaping_Src - set by caller**CUN4BDPR_ArabicOneCellShaping_Targ - set by caller**

Specifies which Arabic one-cell shaping transformations are performed. One-cell shaping refers to the final forms of the seen family.

The effect of this parameter depends on the setting of the `TypeOfText` parameter. Combinations are:

- **ArabicOneCellShaping_Src is TWOCELL_SEEN, and ArabicOneCellShaping_Targ is ONECELL_SEEN, and TypeOfText_Src is TEXT_VISUAL, and TypeOfText_Targ is logical:** Transformation from visual to logical converts final forms of the seen family represented by two characters (the three quarters shape and the tail character) to corresponding nominal code points represented by one character and a space replacing the tail. This space is positioned next to the seen character.
- **ArabicOneCellShaping_Src is ONECELL_SEEN, and ArabicOneCellShaping_Targ is TWOCELL_SEEN, and TypeOfText_Src is logical, and TypeOfText_Targ is TEXT_VISUAL:** In transformation from logical to visual, each character in the seen family which is to receive a final form is converted to the corresponding final form of the seen family that is represented by two characters, consuming an existing space next to the seen character. If there is no space available, it will be converted to one character only which is the three quarters shape seen.
- **Other settings:** Seen tail characters remain as is.

CUN4BDPR_WordBreak_Src - set by caller

CUN4BDPR_WordBreak_Targ - set by caller

Specifies if the service is to transform each word in isolation from adjacent words based on whitespace delimiters.

Combinations are:

- **WordBreak_Src is NO_BREAK, and WordBreak_Targ is BREAK:** Transform each word in isolation from adjacent words based on whitespace delimiters.
- **Other settings:** Do not transform each word in isolation from adjacent words based on whitespace delimiters.

CUN4BDPR_LamAlefEditMode_Src - set by caller

CUN4BDPR_LamAlefEditMode_Targ - set by caller

Specifies which Lam-Alef edit mode transformations are performed.

Combinations are:

- **LamAlefEditMode_Src is LamAlefOff, and LamAlefEditMode_Targ is LamAlefOff:**
 - When transforming from visual to logical layouts, Lam-Alef characters are expanded to Lam plus Alef consuming an existing blank space next to it. If no blank space is available, the Lam-Alef character remains as is.
 - When transforming from logical to visual layouts, Lam plus Alef sequences are compressed to a unique Lam-Alef character; the space resulting from the Lam-Alef compression is positioned next to each generated Lam-Alef character.
- **LamAlefEditMode_Src is LamAlefOff, and LamAlefEditMode_Targ is LamAlefOn:**
 - When transforming from visual to implicit layouts, Lam-Alef characters are expanded to Lam plus Alef consuming a blank space at the absolute end of the buffer. If no blank space is available, the Lam-Alef character remains as is.
 - When transforming from implicit to visual layouts, Lam plus Alef sequences are compressed to a unique Lam-Alef character; the space resulting from Lam-Alef compression is positioned at the absolute end of the buffer.
- **LamAlefEditMode_Src is LamAlefOff, and LamAlefEditMode_Targ is LamAlefAuto:** For each LAMALEF character found, expand LAMALEF using space at end. If there is no space at end, use spaces at beginning of the buffer. If there is no space at the beginning of the buffer, use spaces at the near (for example, the space after the LAMALEF character).
- **Other settings:** Lam Alef characters remain as is.

CUN4BDPR_YehHamzaMode_Src - set by caller

CUN4BDPR_YehHamzaMode_Targ - set by caller

Specifies which YehHamza edit mode transformations are performed. Possible values are:

- **ONECELL_YAHHAMZA**: The Yeh-Hamza final form is represented as one character.
- **TWOCELL_YAHHAMZA**: The Yeh-Hamza final form is represented as two characters.

The default value for CUN4BDPR_YehHamzaMode is TWOCELL_YAHHAMZA, if the CCSID is 00420 or 00864. Otherwise, it is ONECELL_YAHHAMZA.

CUN4BDPR_TailEditMode_Src - set by caller

CUN4BDPR_TailEditMode_Targ - set by caller

Specifies which Tail edit mode transformations are performed. Possible values are:

- **NEW_TAIL**: A newly defined Tail character (U+FE73) in Unicode 3.2 to replace the legacy Seen family Tail character.
- **OLD_TAIL**: A legacy Seen family tail character (U+200B).

The default value for CUN4BDPR_TailEditMode is OLD_TAIL.

CUN4BDPR_TashkeelEditMode_Src - set by caller

CUN4BDPR_TashkeelEditMode_Targ - set by caller

Specifies which Tashkeel edit mode transformations are performed. Possible values are:

- **TASHKEELBEGIN**: All Tashkeel characters (except for Shadda) are replaced by spaces. The resulting spaces are moved to the beginning of the buffer.
- **TASHKEELEND**: All Tashkeel characters (except for Shadda) are replaced by spaces. The resulting spaces are moved to the end of the buffer.
- **TASHKEELREPLACEWITHTATWEEL**: All Tashkeel characters (except for Shadda) are ignored and resize the data buffer. This is done only when the output codepage is 420 or 864.
- **TASHKEELRESIZE**: All Tashkeel characters (except for Shadda) are ignored and resize the data buffer. This is done only when the output codepage is 420 or 864.
- **TASHKEELISOLATED**: All Tashkeel or Tatweel characters (except for Shadda) are ignored and resize the data buffer.

The default value for CUN4BDPR_TashkeelEditMode is TASHKEELEND.

CUN4BDPR_InpToOut_Ptr - set by caller

Specifies a buffer to receive a cross-reference from each Src_Buf code element to the transformed data. The cross-reference relates to the data in Src_Buf starting with the first element that InpBufIndex points to (and not necessarily starting from the beginning of the Src_Buf).

If not a NULL pointer, it points to an array of values with the same number of bytes in Src_Buf starting with the one pointed by InpBufIndex and up to the end of the substring in the buffer. On output, the *n*th value in InpToOut corresponds to the *n*th byte in Src_Buf. This value is the index (in units of bytes) in Targ_Buf that identifies the transformed element of the *n*th byte in Src_Buf. In the case of multibyte encoding, the index points (for each of the bytes of a code element in the Src_Buf) to the first byte of the transformed code element in the Targ_Buf.

InpToOut may be specified as NULL if no index array from Src_Buf to Targ_Buf is desired.

CUN4BDPR_OutToInp_Ptr - set by caller

Specifies a buffer to receive a cross-reference from each Targ_Buf code element to the source buffer. The cross-reference relates to the data in Src_Buf starting with the first element that InpBufIndex points to (and not necessarily starting from the beginning of the Src_Buf).

If not a NULL pointer, it points to an array of values with the same number of bytes in Targ_Buf. On output, the *n*th value in OutToInp corresponds to the *n*th byte in Targ_Buf. This value is the index (in units of bytes) in Src_Buf that identifies the source of the transformed element of the *n*th byte in Targ_Buf. In the case of multibyte encoding, the index points (for each of the bytes of a code element in the Targ_Buf) to the first byte of the source of the transformed code element in the Src_Buf.

OutToInp may be specified as NULL if no index array from Targ_Buf to Src_Buf is desired.

CUN4BDPR_BidiLvl_Ptr - set by caller

A weighted value that represents peculiar input string transformation properties with different connotations as explained below.

If this argument is not a NULL pointer, it represents an array of values with the same number of elements as the Src_Buf before the transformation. Each byte will contain relevant BidiLvl information of the corresponding element in Src_Buf starting from the element pointed by InpBufIndex. The four rightmost bits of each BidiLvl byte will contain information for bidirectional environments (when ActiveDirectional is true) and they will mean NestingLevels. The possible value from 0 to 15 represents the nesting level of the corresponding element in the Src_Buf starting from the element pointed by InpBufIndex. If ActiveDirectional is false, the content of NestingLevel bits will be ignored. The leftmost bit of each BidiLvl byte will contain a new cell indicator for composed character environments and will have a value of either 1 (for an element in Src_Buf that is transformed to the beginning of a new cell) or zero (for the zero-length composing character elements, when these are grouped into the same presentation cell with a non-composing character). Each element of BidiLvl pertains to the elements in the Src_Buf starting from the element pointed by InpBufIndex. Remember that this is not necessarily the beginning of SrcBuf.

If none of the transformation properties is required, the argument property can be NULL.

The use of BidiLvl can be enhanced in the future to pertain to other possible usage in other environments.

CUN4BDPR_InpBufIndex - set by caller, updated by service

InpBufIndex is an offset value to the location of the transformed text. When the bidi service is invoked, InpBufIndex contains the offset to the element in Src_Buf that will be transformed first. Note: This is not necessarily the first element in Src_Buf. At the return from the transformation, InpBufIndex contains the offset to the first element in the Src_Buf that has not been transformed. If the entire substring has been transformed successfully, InpBufIndex will be incremented by the amount defined by Src_Buf_Len.

Set by caller. The service updates the offset value.

CUN4BDPR_Streaming_Processed_Length - set by service

Specifies the amount of source text, in bytes, that layout streaming processed. Set by service when Layout_Streaming is set.

CUN4BDPR_Layout_Streaming_State - set by caller, updated by service

Contains the state of the bidi transformation between calls to the service when Layout_Streaming is used.

The caller should set this area to all zero bytes the first time calling the service with Layout_Streaming and then not modify the value for subsequent calls to the service that use the same layout streaming operation. When using layout streaming, the last call in the sequence is with the Layout_Streaming bit turned off. The caller should not modify the content of the Layout_Streaming_State until after that call returns.

Set by caller and updated by the service when Layout_Streaming is used. Ignored when Layout_Streaming is not used.

CUN4BDPR_Bidi_Keyword - set by caller

This is a short form for extended bidi settings.

Note: Short path settings have higher priority over defaults and long path settings.

Format of CUN4BDPR_Bidi_Keyword:

```
Key1+Value_Key2+Value_Key3+Value...
```

Note:

1. Since most attributes (except for LayoutOptions and CheckMode attributes) can apply to both the source and target data, the second letter in the *key* indicates whether the attributes is for the source (S) or target (T) buffer.

2. If the same key is specified more than once, the last specified value is used.

In the example:

```
0S0_0T1_TS1_TT2
```

- Orientation of the source buffer is LTR.
- Orientation of the target buffer is RTL.
- Type of text of the source buffer is implicit.
- Type of text of the target buffer is explicit.

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|----------------|--|---------------------------------|---|
| LayoutOptions | Lx | 0-252 | <p>Layout options. Values:</p> <ul style="list-style-type: none"> • 1... (128) = CUNBDPRM_Layout_Roundtrip • .1.. (64) = CUNBDPRM_Layout_WinCompat • ..1. (32) = CUNBDPRM_Layout_ImpToImp • ...1 (16) = CUNBDPRM_Layout_Remove_Marks • 1... (8) = CUNBDPRM_Layout_Insert_Marks •1.. (4) = CUNBDPRM_Layout_Streaming <p>Example of Roundtrip and ImpToImp (or Logical to Logical):</p> <pre>L160</pre> <p>For long path equivalent setting, see CUN4BDPR_Layout_Options description.</p> |
| Orientation | Obx | 0-4 | <p>The direction of the text. Values:</p> <ul style="list-style-type: none"> • 0 = ORIENTATION_LTR (Input/Output Default) • 1 = ORIENTATION_RTL • 2 = ORIENTATION_TTBRL • 3 = ORIENTATION_TTBRL • 4 = ORIENTATION_CONTEXTUAL <p>The mappings between short form and long form are defined by BIDI_ORIENTATION in the interface definition file CUNBCIDF.</p> |

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|-----------------------|--|--|--|
| Context | Cbx | 0-1 | <p>Contextual orientation when the orientation attribute is set to ORIENTATION_CONTEXTUAL. Values:</p> <ul style="list-style-type: none"> • 0 = CONTEXT_LTR (Input/Output Default) • 1 = CONTEXT_RTL <p>The mappings between short form and long form are defined by BIDI_CONTEXT in the interface definition file CUNBCIDF.</p> |
| TypeofText | Tbx | 0-2 | <p>Type of the text. Values:</p> <ul style="list-style-type: none"> • 0 = TEXT_VISUAL (Output default) • 1 = TEXT_IMPLICIT (Input default) • 2 = TEXT_EXPLICIT <p>The mappings between short form and long form are defined by BIDI_TEXT_TYPE in the interface definition file CUNBCIDF.</p> |
| ImplicitAlg | Ibx | 0-1 | <p>Implicit algorithm used in the source/target buffer. Values:</p> <ul style="list-style-type: none"> • 0 = ALGOR_BASIC (Input/Output Default) • 1 = ALGOR_IMPLICIT <p>The mappings between short form and long form are defined by BIDI_IMPALG in the interface definition file CUNBCIDF.</p> |
| Swapping | Sbx | 0-1 | <p>Specifies whether symmetric swapping is enabled. Values:</p> <ul style="list-style-type: none"> • 0 = SWAPPING_NO (Output default) • 1 = SWAPPING_YES (Input default) <p>The mappings between short form and long form are defined by BIDI_SWAPPING in the interface definition file CUNBCIDF.</p> |
| Numerals | Nbx | 0-3 | <p>How numerals are shaped. Values:</p> <ul style="list-style-type: none"> • 0 = NUMERALS_NOMINAL (Input default. Output default in Hebrew locale.) • 1 = NUMERALS_NATIONAL • 2 = NUMERALS_CONTEXTUAL (Output default in Arabic locale) • 3 = NUMERALS_NONE <p>The mappings between short form and long form are defined by BIDI_NUMERALS in the interface definition file CUNBCIDF.</p> |

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|----------------|--|---------------------------------|--|
| TextShaping | Ebx | 0-7 | <p>Specifies whether text to be shaped. Values:</p> <ul style="list-style-type: none"> • 0 = TEXT_SHAPED (Output default in Arabic locale) • 1 = TEXT_NOMINAL (Input default, Output default in Hebrew locale) • 2 = TEXT_SHFORM1 • 3 = TEXT_SHFORM2 • 4 = TEXT_SHFORM3 • 5 = TEXT_SHFORM4 • 6 = TEXT_STANDARD • 7 = TEXT_COMPOSED <p>The mappings between short form and long form are defined by BIDI_SHAPING in the interface definition file CUNBCIDF.</p> |
| CheckMode | Hx | 0-1 | <p>Level of Bidi checking (apply to both source and target). Values:</p> <ul style="list-style-type: none"> • 0 = MODE_STREAM • 1 = MODE_EDIT (Input/Output default) <p>The mappings between short form and long form are defined by BIDI_CHECKMODE in the interface definition file CUNBCIDF.</p> |
| WordBreak | Wbx | 0-1 | <p>Word break. Values:</p> <ul style="list-style-type: none"> • 0 = WORD_BREAK • 1 = NO_BREAK (Input/Output default) <p>The mappings between short form and long form are defined by BIDI_WORDBREAK in the interface definition file CUNBCIDF.</p> |
| LamAlefEdit | Fbx | 0-5 | <p>LamAlef edit mode. Values:</p> <ul style="list-style-type: none"> • 0 = LamAlefOn • 1 = LamAlefBegin • 2 = LamAlefResize • 3 = LamAlefNear • 4 = LamAlefAuto (Input/Output default) • 5 = LamAlefOff <p>The mappings between short form and long form are defined by BIDI_LAMALEF in the interface definition file CUNBCIDF.</p> |

| Attribute name | Format key: b=buffer (S=source or T=target) x=attribute value | Possible attribute values | Description |
|----------------|--|---------------------------------|--|
| ArabicOneCell | Abx | 0-1 | Arabic one-cell shaping. Values: <ul style="list-style-type: none"> 0 = ONECELL_SEEN (Input default. Output default for Hebrew locale.) 1 = TWOCELL_SEEN (Output default for Arabic locale.) The mappings between short form and long form are defined by BIDI_ONECELL in the interface definition file CUNBCIDF. |
| TailMode | Mbx | 0-1 | Tail edit mode. Values: <ul style="list-style-type: none"> 0 = NEW_TAIL 1 = OLD_TAIL The mappings between short form and long form are defined by BIDI_TAIL in the interface definition file CUNBCIDF. |
| TashkeelMode | Kbx | 0-4 | Tashkeel edit mode. Values: <ul style="list-style-type: none"> 0 = TashkeelBegin 1 = TashkeelEnd 2 = TashkeelReplaceWithTatweel 3 = TashkeelResize 4 = TashkeelIsolated The mappings between short form and long form are defined by BIDI_TASHKEEL in the interface definition file CUNBCIDF. |
| YehHamza | Ybx | 0-1 | YehHamza edit mode. Values: <ul style="list-style-type: none"> 0 = ONECELL_YEHHAMZA (Input default. Output default for Hebrew locale.) 1 = TWOCELL_YEHHAMZA (Output default for Arabic locale.) The mappings between short form and long form are defined by BIDI_YEHHAMZA in the interface definition file CUNBCIDF. |

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:

| <i>Table 6. Minimum and maximum character widths of the different encoding schemes</i> | | | | |
|--|--------------|-------------------------|-------------------------|--|
| Encoding scheme | ESID | Minimum Character Width | Maximum Character Width | Rationale |
| SBCS | x1xx | 1 | 1 | pure single byte |
| 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. In that case, 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: $(1 \text{ or } 2) + 4 + 1 = (6 \text{ or } 7)$. |

| Table 6. Minimum and maximum character widths of the different encoding schemes (continued) | | | | |
|---|------|-------------------------|-------------------------|---|
| Encoding scheme | ESID | Minimum Character Width | Maximum Character Width | Rationale |
| 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.

Using the piece-by-piece method is not recommended when using the B technique. The B technique requires complete input to get correct results. You can use the piece-by-piece technique when using the extended bidi support with Layout_Streaming.

Sample programs

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

- CUNSCSMC for C.
- CUNSCSMA for HLASM.
- CUNISM7 shows how to invoke the extended bidi support using C.
- CUNISM8 shows how to implement the Open Group's bidi support using Unicode Services extended bidi support.

Chapter 4. Case conversion

This information 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 Services supports the following types of casing:

- Upper mapping:
 - Simple case to upper mapping (where string lengths do not change) based on Unicode data file only.
 - Special case to upper mapping (where string lengths may change) with or without additional information provided by a locale.
- Lower mapping:
 - Simple case to lower mapping (where string lengths do not change) based on Unicode data file only.
 - Special case to lower mapping (where string lengths may change) with or without additional information provided by a locale.
- Title case:
 - Simple title case (where string lengths do not change).
 - Special title case mapping (where string lengths may change) with or without additional information provided by a locale.
- INITCAP case:
 - Simple INITCAP case (where string lengths do not change).
 - Special title case mapping (where string lengths may change) with or without additional information provided by a locale.

The INITCAP function returns a string that the first character of each word would be converted to uppercase and the other characters would be converted to lowercase. Words are delimited by a set of the following characters:

| <i>Table 7. Word delimiter characters</i> | |
|--|---|
| Character or range of characters | Unicode code points or range of Unicode code points |
| (blank) | U+0020 |
| !"#\$%&'()*+,-./ | U+0021 to U+002F |
| ;<=>?@ | U+003A to U+0040 |
| [\]^_` | U+005B to U+0060 |
| { }~ | U+007B to U+007E |
| Control characters, including the following SQL control characters: <ul style="list-style-type: none"> – tab – new line – form feed – carriage return – line feed | U+0009, U+000A, U+000B, U+000C, U+000D, U+0085 |

With this function, the following case conversion of strings can be implemented:

- Convert string from 'a prospective book title' to 'A Prospective Book Title'
- Convert string from 'YOUR NAME' to 'Your Name'
- Convert string from 'my_résumé' to 'My_Résumé'
- Convert string from 'FORMAT:élégant' to 'Format:Élégant'

Unicode case conversion is described in the Unicode standard at [The Unicode Consortium \(www.unicode.org\)](http://www.unicode.org). Case conversion rules are summarized in the two tables **UnicodeData.txt** and **SpecialCasing.txt** which are available from the same website.

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 254](#) and [“Case conversion” on page 264](#).

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).

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 8. 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 81](#). 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 */
```

Case conversion

```
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 - set by the service */
        /* Reserved */
    } Flag2;
    unsigned char Res5[2];
    long Return_Code;
    long Reason_Code;
    unsigned char Res6[3]; /* Reserved */
    unsigned char UniVersion; /* Unicode Data version */
} CUNBAPRM;
```

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 */
            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; /* FLAG1; */
        /* FLAG Byte 1 set by caller */
    }
} 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) */
/* Padding */
: 7; /* Flag2 - set by the service */
/* Reserved */
} Flag2;
unsigned char Res5[2];
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.
                Save parameter area address
LR      R4,R1    Make parameter area addressable
USING   CUNBAPRM,R4 Init PARAMETER AREA TO BINARY 0
XC      CUNBAPRM,CUNBAPRM
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
                Save parameter area address
LR      R4,R1    Make parameter area addressable
USING   CUN4BAPR,R4 Init PARAMETER AREA TO BINARY 0
XC      CUN4BAPR,CUN4BAPR
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
*                and length).

```

* 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 9. 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 |
| 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 | DWORD | 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 |
| 156 | (9C) | 1... .. | 1 | | CUNBAPRM_Inv_Handle | Invalid handle action: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONTINUE. |
| 156 | (9C) | .1... .. | 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. |
| 156 | (9C) | ..1. | 1 | | CUNBAPRM_Page_Fix | Page fixing: 0=System storage 1=Page Fixing |

| Table 9. 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 |
| 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 |
| 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.

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:

| Constant | Description |
|-----------------------|--|
| 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 |
| CUNBAPRM_To_INITCAP | Converts to INITCAP case, includes simple casing only. |
| CUNBAPRM_To_INITCAP_S | Converts to INITCAP case, includes locale independent special casing. |

| Constant | Description |
|-----------------------|---|
| CUNBAPRM_To_INITCAP_L | Converts to INITCAP case, includes locale dependent special casing. |

Conversion types CUNBAPRM_To_Title, CUNBAPRM_To_Title_S and CUNBAPRM_To_Title_L, CUNBAPRM_To_INITCAP, CUNBAPRM_To_INITCAP_S and CUNBAPRM_To_INITCAP_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, CUNBAPRM_To_Title_L or CUNBAPRM_To_INITCAP_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.

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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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 [“Locales supported for case service”](#) on page 554.
- **Locale NOT supported:** CUNBAPRM_Locale_Support content matches one of the locale names for Case service support list (See [“Locales supported for case service”](#) on page 554).

| 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. |

| Terms | CUNBAPRM_Locale_Support Value | Description |
|---------------|-------------------------------|--|
| NOT supported | 1 | <p>When Return/Reason Code is not set to</p> <div>CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP</div> <p>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.</p> |

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.
- CUNBAPRM_UNI600, 6.0.0 Unicode data version is requested.
- CUNBAPRM_UNI900, 9.0.0 Unicode data version is requested.
- CUNBAPRM_UNIX13, 13.0.0 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 10. 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 | 192 | DWORD | CUN4BAPR | Parameter Area |
| 0 | (0) | UNSIGNED | 4 | | CUN4BAPR_Version | Parameter Area VERSION |
| 4 | (4) | UNSIGNED | 4 | | CUN4BAPR_Length | Parameter area Length |

Table 10. 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 |
|------------|------------|-----------|-----------------|----------|------------------------|---|
| 8 | (8) | ADDRESS | 8 | | CUN4BAPR_Src_Buf_Ptr | Source buffer pointer |
| 16 | (10) | UNSIGNED | 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) | ADDRESS | 8 | | CUN4BAPR_Targ_Buf_Ptr | Target buffer pointer |
| 40 | (28) | UNSIGNED | 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 | 7 | | * | Reserved |
| 128 | (80) | CHARACTER | 32 | | CUN4BAPR_Locale | Language locale used for case conversion |
| 160 | (A0) | ADDRESS | 8 | DWORD | CUN4BAPR_DDA_Buf_Ptr | Dynamic data area pointer |
| 168 | (A8) | UNSIGNED | 4 | | CUN4BAPR_DDA_Buf_ALET | Dynamic data area ALET |
| 172 | (AC) | UNSIGNED | 4 | | CUN4BAPR_DDA_Buf_Len | Dynamic data area length as defined by constant CUN4BAPR_DDA_Req. |
| 176 | (B0) | BITSTRING | 1 | | CUN4BAPR_Flag1 | FLAG Byte 1 set by caller |
| 176 | (B0) | 1... .. | 1 | | CUN4BAPR_Inv_Handle | Invalid handle action: 0=TERMINATE WITH ERROR. 1=GET NEW HANDLE AND CONT. |
| 176 | (B0) | .1... .. | 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. |
| 176 | (B0) | ..1. | 1 | | CUN4BAPR_Page_Fix | Page fixing: 0=System storage 1=Page Fixing |
| 177 | (B1) | UNSIGNED | 1 | | CUN4BAPR_Flag2 | FLAG Byte 2 (Set by caller) |

Table 10. 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) |
| 178 | (B2) | CHARACTER | 2 | | * | Reserved |
| 180 | (B4) | CHARACTER | 8 | WORD | CUN4BAPR_RC_RS | Return/reason code |
| | | UNSIGNED | 4 | | CUN4BAPR_Return_Code | Return code |
| | | UNSIGNED | 4 | | CUN4BAPR_Reason_Code | Reason code |
| 188 | (BC) | CHARACTER | 3 | | * | Reserved |
| 191 | (BF) | CHARACTER | 1 | | CUN4BAPR_UniVersion | Unicode Data Version |
| 192 | (C0) | | 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.

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:

| Constant | Description |
|-----------------------|--|
| 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 |
| CUN4BAPR_To_INITCAP | Converts to INITCAP case, includes simple casing only. |
| CUN4BAPR_To_INITCAP_S | Converts to INITCAP case, includes locale independent special casing. |

| Constant | Description |
|-----------------------|---|
| CUN4BAPR_To_INITCAP_L | Converts to INITCAP case, includes locale dependent special casing. |

Conversion types CUN4BAPR_To_Title, CUN4BAPR_To_Title_S, CUN4BAPR_To_Title_L, CUN4BAPR_To_INITCAP, CUN4BAPR_To_INITCAP_S and CUN4BAPR_To_INITCAP_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, CUN4BAPR_To_Title_L or CUN4BAPR_To_INITCAP_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.

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.

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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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 [“Locales supported for case service”](#) on page 554 topic.
- **Locale NOT supported:** CUN4BAPR_Locale content matches one of the locale names for the case service support (See [“Locales supported for case service”](#) on page 554).

| 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. |

| Terms | CUN4BAPR_Locale_Support Value | Description |
|---------------|-------------------------------|--|
| NOT supported | 1 | <p>When Return/Reason Code is <i>not</i> set to</p> <div>CUN_RC_USER_ERR/ CUN_RS_CASE_NOT_SUPP</div> <p>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.</p> |

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.
- 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.
- CUN4BAPR_UNI600, 6.0.0 Unicode data version is requested.
- CUN4BAPR_UNI900, 9.0.0 Unicode data version is requested.
- CUN4BAPR_UNIX13, 13.0.0 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:

- CUN4A01C for C
- CUN4A02A for HLASM

Chapter 5. Normalization

This topic 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 Standard input string. Normalization is described in *Unicode Normalization Forms*, which is available in the [Unicode Technical Reports \(www.unicode.org/reports\)](http://www.unicode.org/reports).

Normalization rules are based on the Unicode Standard versions listed in [Table 11 on page 97](#). Each standard can be accessed from the [Unicode Public Directory \(www.unicode.org/Public\)](http://www.unicode.org/Public).

| Table 11. Unicode version table | |
|---------------------------------|--|
| Unicode version | Directory location and file name |
| UNI301 | 3.0-Update1/UnicodeData-3.0.1.txt |
| UNI301 | 3.0-Update1/CompositionExclusions-2.txt |
| UNI320 | 3.2-Update/UnicodeData-3.2.0.txt |
| UNI320 | 3.2-Update/CompositionExclusions-3.2.0.txt |
| UNI401 | 4.0-Update1/UnicodeData-4.0.1.txt |
| UNI401 | 4.0-Update/CompositionExclusions-4.0.0.txt |
| UNI410 | 4.1.0/ucd/UnicodeData.txt |
| UNI410 | 4.1.0/ucd/CompositionExclusions.txt |
| UNI600 | 6.0.0/ucd/UnicodeData.txt |
| UNI600 | 6.0.0/ucd/CompositionExclusions.txt |
| UNI900 | 9.0.0/ucd/UnicodeData.txt |
| UNI900 | 9.0.0/ucd/CompositionExclusions.txt |
| UNIX13 | 13.0.0/ucd/UnicodeData.txt |
| UNIX13 | 13.0.0/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 254](#) and [“Normalization conversion” on page 265](#). 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)

Normalization

- Flags
- Unicode Version

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 12. 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 81](#). 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;
        unsigned char Res6[3]; /* FLAG Byte 1 set by caller */
        /* 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 */

```

Normalization

```
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   */
    Page_Fix      : 1, /* Page Fixing:              */
/* 0 = System storage      */
/* 1 = Page Fixing         */
    : 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;
```

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.

Table 13. 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 |

Table 13. 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 |
|------------|------------|-----------|-----------------|----------|-----------------------|--|
| 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 | DWORD | 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 |
| 144 | (90) | 1... .. | 1 | | CUNBNPRM_Inv_Handle | Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONTINUE. |
| 144 | (90) | .1... .. | 1 | | CUNBNPRM_Page_Fix | Page Fixing: 0=System storage management (default). 1=Page fixing. |
| 144 | (90) | ..11 1111 | 1 | | * | 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 |
| 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, 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):

| Constant | Description |
|-------------|--|
| 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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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.
- CUNBNPRM_UNI600, 6.0.0 Unicode data version is requested.
- CUNBNPRM_UNI900, 9.0.0 Unicode data version is requested.
- CUNBNPRM_UNIX13, 13.0.0 Unicode data version is requested.

Mapping of parameters for AMODE (64)

The mapping of the parameter areas is supplied by the interface definition file 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.

| Table 14. 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 | DWORD | 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 |

Table 14. 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 |
|------------|------------|-----------|-----------------|----------|----------------------|--|
| 168 | (A8) | 1... .. | 1 | | CUN4BNPR_Inv_Handle | Invalid handle at start: 0=TERMINATE WITH ERROR 1=GET NEW HANDLE AND CONTINUE. |
| 168 | (A8) | .1... .. | 1 | | CUN4BNPR_Page_Fix | Page Fixing: 0=System storage management (default). 1=Page fixing. |
| 168 | (A8) | ..11 1111 | 1 | | * | 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 |
| 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, 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):

| Constant | Description |
|-------------|--|
| 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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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.
- CUN4BNPR_UNI600, 6.0.0 Unicode data version is requested.
- CUN4BNPR_UNI900, 9.0.0 Unicode data version is requested.
- CUN4BNPR_UNIX13, 13.0.0 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

64-bit samples:

- CUN4SNSC for C
- CUN4SNSA for HLASM

Chapter 6. Collation

This topic 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 z/OS 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.

Collation supports customization, which means that collation service might behave according to some specific collation rules. Collation rules can be specified using a Locale or User Collation Rules (UCR). The following are the collation versions:

UCA301

This collation version supports the Unicode Standard character suite 3.0.1 and does not support customization.

UCA400R1

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

UCA410

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

UCA600

This collation version supports the Unicode Standard character suite 6.0.0 and uses Normalization Service under 6.0.0 Unicode character suite.

UCA900

This collation version supports the Unicode Standard character suite 9.0.0 and uses Normalization Service under 9.0.0 Unicode character suite.

UCAX13

This collation version supports the Unicode Standard character suite 13.0.0 and uses Normalization Service under 13.0.0 Unicode character suite.

This z/OS Unicode Services 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, 4.1.0, 6.0.0, 9.0.0, and 13.0.0. For further information about the Unicode Standard collation standard, see *Unicode Collation Algorithm* in the [Unicode Technical Reports \(www.unicode.org/reports\)](http://www.unicode.org/reports).

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 Services 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 131](#)). 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. Collation version 6.0.0 requires normalization service 6.0.0 which is supported by z/OS V2R1 and later. Collation version 9.0.0 requires normalization service 9.0.0 which is supported by z/OS V2R3 and later. Collation version 13.0.0 requires normalization service 13.0.0 which is supported by z/OS V2R5 and later. See [Chapter 5, “Normalization,” on page 97](#) for more information.

For detailed information, see [“Creating a conversion image” on page 254](#) and [“Collation conversion” on page 265](#).

Calling the collation service

This topic describes how the z/OS support for the Unicode Standard 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 collation features (UCA400R1 and higher), 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 must always provide the following, regardless of which of the three cases 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 131](#).

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 higher 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 165](#) and [“Work buffer length considerations” on page 164](#).

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 higher, 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 (www.unicode.org). Collation version 3.0.1 follows sort key generation as described on the Unicode Consortium TR#10, while recent Collation versions UCA400R1 and higher 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 163](#). 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 Collation versions UCA400R1 and higher, sort key weights have different values than their respective versions from the DUCET (Default Unicode Collation Element Table (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 131. 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.

| <i>Table 15. Restrictions for the calling environment</i> | |
|---|---|
| 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 120.

```

/* 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] = {
/* HELLO */
/* ----- */
'\x00', '\x48', '\x00', '\x45', '\x00', '\x4C', '\x00', '\x4C', '\x00', '\x4F'
};
unsigned char Workbuffer1 [WLEN];
unsigned char Targetbuffer1 [TLEN];
/* Group 2                                            */
unsigned char Sourcebuffer2 [SLEN] = {

```



```

/* HELLO */
/* -----
'\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 */
/* 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 ] = {
/* HELLO */
/* -----
'\x00','\x48','\x00','\x45','\x00','\x4C','\x00','\x4C','\x00','\x4F'
};
unsigned char Workbuffer1 [WLEN ];
unsigned char Targetbuffer1 [TLEN ];
/* Group 2 */
unsigned char Sourcebuffer2 [SLEN ] = {
/* HELLO */
/* -----
'\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 */
  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 ] = {
/* HELLO */
/* ----- */
'\x00','\x48','\x00','\x45','\x00','\x4C','\x00','\x4C','\x00','\x4F'
};
unsigned char Workbuffer1 [WLEN ];
unsigned char Targetbuffer1 [TLEN ];
/*          Group 2          */
unsigned char Sourcebuffer2 [SLEN ] = {
/* HELLO */
/* ----- */
'\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      O O O
/*      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, O, D |
/* +-----+
/* |Case_First |C | X, L, U, D |
/* +-----+
/* |Alternate |A | N, S, D |
/* +-----+
/* |Variable_Top |T |
/* +-----+
/* |Normalization |N |X, O, D |
/* +-----+
/* |French |F |X, O, D |
/* +-----+
/* |Hiragana |H |X, O, 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
    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;
    } Flag1;
    unsigned char Res9[1]; /* FLAG Byte 1 set by caller */
    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;
    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, /* Padding */
                        /* 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 tagCUN4B0PR {
    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 */
    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 */
        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;                /* FLAG Byte 1 set by caller */
    } Flag1;
    unsigned char   Res8;              /* 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 */
        int         SKey_and_Cmp        : 1, /* Make binary comparison */
                                           /* (CUNBOPRM_RESULT), if and */
                                           /* only if, CUN4B0PR_SKey_Opt */
                                           /* is ON: */
                                           /* 0- Do not perform binary */
                                           /* comparison */
                                           /* 1- Perform binary comparison */
        int         : 7;                /* Padding */
    } Coll_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     */
                                           /* Where:                */
                                           /* 0 - Default           */
                                           /* 1 - Primary Level will */
                                           /* ignore accent but not */
                                           /* case                  */
                                           /* Padding              */
    int         Case_Level           : 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                 */
                                           /* Reserved              */
    : 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:

```

#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_UCAempty '\x00'
#define CUNBOPRM_UCA301 '\x01'
#define CUNBOPRM_UCA400R1 '\x02'
#define CUNBOPRM_UCA410 '\x03'
#define CUNBOPRM_UCA600 '\x04'
#define CUNBOPRM_UCA900 '\x05'
#define CUNBOPRM_UCAX13 '\x06'
```

Using the HLASM interface

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.

Chapter 6. Collation **125**

```

*****
*                               /*****
*                               /*    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,STR1LEN         ! 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      */
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 166.

Mapping of parameters for AMODE (31)

The mapping of the parameter areas is supplied by the interface definition file CUNBODIF. 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 16. 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 |
| 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 | DWORD | CUNBOPRM_DDA_Buf_Ptr | Dynamic data area pointer |
| 184 | (B8) | UNSIGNED | 4 | | CUNBOPRM_DDA_Buf_ALET | Dynamic data area ALET |

| Table 16. 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 |
| 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 |
| 192 | (C0) | 1... .. | 1 | | CUNBOPRM_Inv_Handle | Invalid handle action: 0=TERMINATE WITH ERROR. 1=GET NEW HANDLE AND CONT. |
| 192 | (C0) | .1... .. | 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 |
| 192 | (C0) | ..1. | 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 |
| 194 | (C2) | BITSTRING | 1 | | CUNBOPRM_Mask1 | |
| 194 | (C2) | 111. | 1 | | CUNBOPRM_Variable_Opt | Where: 0=Shifted 1=Blanked 10=Non Blanked 11=Shift Trimmed and Reserved |
| 194 | (C2) | ...1 | 1 | | CUNBOPRM_Cmp_Order | Where: 0=Forward 1=Backward (French) |
| 194 | (C2) | 1... | 1 | | CUNBOPRM_Skey_Opt | Where: 0=No get sort key 1=Get sort key |
| 194 | (C2) |111 | 1 | | 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 | |

Table 16. 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 |
|------------|------------|-----------|-----------------|----------|-----------------------------|---|
| 195 | (C3) | 1... .. | 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 |
| | | 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 |

| Table 16. 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 |
| 220 | (DC) | CHARACTER | 32 | | CUNBOPRM_Locale | Locale Input (ll_CC_Variant) |
| | | CHARACTER | 2 | | CUNBOPRM_Locale_ll | 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 Collation features (UCA versions UCA400R1 and higher, 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 165](#) and [“Sort key vector format” on page 163](#).

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 165](#) 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 165](#) and [“Sort key vector format” on page 163](#).

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 165](#) 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 164 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 164 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 164 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 164 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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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 17. Collation mask sub fields descriptions | |
|--|--|
| Sub fields | Description |
| CUNBOPRM_Variable_Opt | <p>This sub field specifies if operations with variable collation elements must be performed. The options are:</p> <ul style="list-style-type: none"> 0 - Shifted (SHIFTED) 1 - Blanked (BLANKED) 2 - Non-Ignored (NIGNORED) 3 - Shift-Trimmed (STRIMMED) 4 - No Variable Behavior (NAVARIABLECE) |

| Table 17. Collation mask sub fields descriptions (continued) | |
|--|---|
| Sub fields | Description |
| CUNBOPRM_Cmp_Order | <p>This sub field specifies following comparison orders:</p> <ul style="list-style-type: none"> 0 - Forward (FORWARD) (Default) 1 - Backward (BACKWARD) (French behavior) |
| CUNBOPRM_SKey_Opt | <p>This sub field specifies either a comparison or sort key:</p> <ul style="list-style-type: none"> 0 - No get sort key (SKOFF) and perform binary comparison. 1 - Get sort key (SKON) and do not perform binary comparison. |
| CUNBOPRM_Norm_Type | <p>This sub field specifies the normalization form according to the following values:</p> <ul style="list-style-type: none"> 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 | <p>Perform Binary comparison when Sort Key is also requested.</p> <ul style="list-style-type: none"> 0 - Do not perform binary comparison (default) 1 - perform binary comparison <p>Note: This bit flag will be meaningful if the following flags are set:</p> <ul style="list-style-type: none"> • CUNBOPRM_Version = CUNBOPRM_Ver2 • CUNBOPRM_SKey_Opt = SKON • CUNBOPRM_UCA_Ver = CUNBOPRM_UCA400R1 (or higher) <p>Collation version 3.0.1, was able to generate either:</p> <ul style="list-style-type: none"> • Perform Binary comparisons or • Generate Sort Key <p>But not both.</p> <p>From UCA400R1 and higher, its possible to generate sort key and perform binary comparison at the same time.</p> |

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 higher, 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 higher, 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 higher, 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 higher. 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 E, “Locales for collation and case support,” on page 537.](#)
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 higher, otherwise its content will be ignored.

Unicode Locales repository data set name SYS1.SCUNLOCL contains a set of locales documented in [Appendix E, “Locales for collation and case support,” on page 537.](#) 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"                             *
* *                                                         *
* (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 E, “Locales for collation and case support,” on page 537.](#)

For further information about Collation rules syntax, see CUNBOPRM_Collation_Rules_File field description.

From [Appendix E, “Locales for collation and case support,” on page 537](#) 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 18. 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 Collation callers for UCA400R1 and higher versions in the Collation API. This field can be set according the following table:

Table 19. Collation keywords descriptions

| Attribute Name | Key | Possible Values | Description |
|----------------|-------|------------------|---|
| Locale | L R V | <locale> | <p>Provide a specific locale for collation rules which are in SYS1.SCUNLOCL repository. For Locales supported, see Appendix E, "Locales for collation and case support," on page 537.</p> <p>Where "Attribute Name" has the following format:</p> <p>Lxx_Ryy_Vzz, where:</p> <ul style="list-style-type: none"> • L means language • R means region • V means variant <p>Example:</p> <pre>UCA400R1_LSV (Swedish) "Kyppe" < "Köpfe"</pre> <p>For long path equivalent setting, see CUNBOPRM_Locale description.</p> |
| Strength | S | 1, 2, 3, 4, I, D | <p>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).</p> <p>Example:</p> <pre>UCA400R1_S1 role = Role = rôle UCA400R1_S2 role = Role < rôle UCA400R1_S3 role < Role < rôle</pre> <p>For long path equivalent setting, see CUNBOPRM_Coll_Level description.</p> |

Table 19. Collation keywords descriptions (continued)

| Attribute Name | Key | Possible Values | Description |
|----------------|-----|-----------------|---|
| Case_Level | K | X, O, D | <p>The Case Level attribute is used when ignoring accents but not case. In such case, set Strength to Primary, and Case_Level to On.</p> <p>In most locales, this setting is Off by default.</p> <p>Example:</p> <pre>UCA400R1_S1_KX role = Role = rôle UCA400R1_S1_K0 role = rôle < Role</pre> <p>For long path equivalent setting, see CUNBOPRM_Case_Level description.</p> |
| Case_First | C | X, L, U, D | <p>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.</p> <p>Example:</p> <pre>UCA400R1_CX or UCA400R1_CL "china" < "China" < "denmark" < "Denmark" UCA400R1_CU "China" < "china" < "Denmark" < "denmark"</pre> <p>For long path equivalent setting, see CUNBOPRM_Case_First description.</p> |
| Alternate | A | N, S, D | <p>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.</p> <p>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.</p> <p>If Alternate is not set to Shifted, then there is no difference between a Strength of 3 and a Strength of 4.</p> <p>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.</p> <p>Example:</p> <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> <p>For long path equivalent setting, see CUNBOPRM_Variable_Opt description.</p> |
| Variable_Top | T | <hex digits> | <p>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.</p> <p>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".</p> <p>Example:</p> <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> <p>For long path equivalent setting, see CUNBOPRM_Var_Top description.</p> |

Table 19. Collation keywords descriptions (continued)

| Attribute Name | Key | Possible Values | Description |
|------------------------|-----|-----------------|---|
| Normalization Checking | N | X, O, D | <p>The Normalization setting determines whether text is thoroughly normalized or not in comparison (see also CUN4BOPR_Norm_Type).</p> <p>Example:</p> <pre>UCA400R1_NX ä= a + ï% < ä+ ï% < ;+ ï% UCA400R1_NO ä= a + ï% < ä+ ï% < ;+ ï%</pre> <p>For long path equivalent setting, see CUNBOPRM_Norm_Type description.</p> |
| French | F | X, O, D | <p>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).</p> <p>Example:</p> <pre>UCA400R1_FX cote < coté< côte < côté UCA400R1_F0 cote < côte< coté < côté</pre> <p>For long path equivalent setting, see CUNBOPRM_Cmp_Order description.</p> |
| Hiragana | H | X, O, D | <p>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.</p> <p>Example:</p> <pre>UCA400R1_HX_S4 M0... = -ã< M0†= -0æ UCA400R1_HO_S4 M0... < -ã< M0†< -0æ</pre> <p>For long path equivalent setting, see CUNBOPRM_Hiragana description.</p> |

Valid values for collation keywords are listed in the following table:

| Table 20. 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 one of the following Collation versions followed by desired sets:

- *UCA400R1_...
- *UCA410_...
- *UCA600_...
- *UCA900_...
- *UCAX13_...

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 21. 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 "&ae << æ" is equivalent to "&a << æ/e". |
| | a b | 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 22. 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 the example that follows table "Collation syntax rules". | 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. |
| Strength | [strength 1] [strength 2] [strength 3] [strength 4] [strength 5] [strength I] | 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 in this topic.
- 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 (CUNBOIDF).

```
* *****
* *                               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'     Forward 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
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_UCA600 EQU 4
CUNBOPRM_UCA900 EQU 5
CUNBOPRM_UCAX13 EQU 6
*
* *****
* *                               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 23. 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 |
| 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 |

| Table 23. 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 |
| 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 | DWORD | 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 |
| 240 | (F0) | 1... .. | 1 | | CUN4BOPR_Inv_Handle | Invalid handle action: 0=TERMINATE WITH ERROR. 1=GET NEW HANDLE AND CONT. |
| 240 | (F0) | .1... .. | 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 |
| 240 | (F0) | ..1. | 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 | |
| | | 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 |

Table 23. 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_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 (CUNBOPR_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 |
| | | 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 |

| Table 23. 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_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 (ll_CC_Variant) |
| | | CHARACTER | 2 | | CUN4BOPRM_Locale_ll | 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 CUN4BOID.

In order to exploit new Collation features (UCA versions UCA400R1 and higher, 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 CUN4BOID.

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 165](#) and [“Sort key vector format” on page 163](#).

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 165](#) 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 165](#) and [“Sort key vector format” on page 163](#).

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 165](#) 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 CUN4BOID):

- 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 164](#) 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 164](#) 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 164](#) 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 164 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 (CUN4BOID).

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

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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 24. Collation mask sub fields descriptions | |
|--|---|
| Sub fields | Description |
| CUN4BOPR_Variable_Opt | <p>This sub field specifies if operations with variable collation elements must be performed. The options are:</p> <ul style="list-style-type: none"> 0 - Shifted (SHIFTED) 1 - Blanked (BLANKED) 2 - Non-Ignored (NIGNORED) 3 - Shift-Trimmed (STRIMMED) 4 - No Variable Behavior (NAVARIABLECE) |
| CUN4BOPR_Cmp_Order | <p>This sub field specifies following comparison orders:</p> <ul style="list-style-type: none"> 0 - Forward (FORWARD) (Default) 1 - Backward (BACKWARD) (French behavior) |
| CUN4BOPR_SKey_Opt | <p>This sub field specifies either a comparison or sort key:</p> <ul style="list-style-type: none"> 0 - No get sort key (SKOFF) and perform binary comparison. (Default) 1 - Get sort key (SKON) and do not perform binary comparison. |
| CUN4BOPR_Norm_Type | <p>This sub field specifies the normalization form according to the following values:</p> <ul style="list-style-type: none"> 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 | <p>Perform Binary comparison when Sort Key is also requested.</p> <ul style="list-style-type: none"> 0 - Do not perform binary comparison (default) 1 - perform binary comparison <p>Note: This bit flag will be meaningful if the following flags are set:</p> <ul style="list-style-type: none"> • CUN4BOPR_Version = CUN4BOPR_Ver2 • CUN4BOPR_SKey_Opt = SKON • CUN4BOPR_UCA_Ver = CUN4BOPR_UCA400R1 (or higher) <p>Collation version 3.0.1, was able to generate either:</p> <ul style="list-style-type: none"> • Perform Binary comparisons or • Generate Sort Key <p>But not both.</p> <p>From UCA400R1 and higher, its possible to generate sort key and perform binary comparison at the same time.</p> |

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 higher, 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 higher, 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 CUN4BOPR_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 higher, 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 higher. 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 E, “Locales for collation and case support,” on page 537](#).
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 higher, otherwise its content will be ignored.

Unicode Locales repository data set name SYS1.SCUNLOCL contains a set of locales documented in [Appendix E, “Locales for collation and case support,” on page 537](#). 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"                 *
*                                               *
* (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 E, “Locales for collation and case support,” on page 537](#).

For further information about Collation rules syntax, see CUN4BOPR_Collation_Rules_File field description.

From Appendix E, “Locales for collation and case support,” on page 537 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.

| CUN4BOPR_Collation_Keyword | CUN4BOPR_Locale_Language | CUN4BOPR_Locale_Region | CUN4BOPR_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 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 Collation callers for UCA400R1 and higher versions in the Collation API. This field can be set according the following table:

| Attribute Name | Key | Possible Values | Description |
|----------------|-------|------------------|--|
| Locale | L R V | <locale> | <p>Provide a specific locale for collation rules which are in SYS1.SCUNLOCL repository. For Locales supported, see Appendix E, “Locales for collation and case support,” on page 537.</p> <p>Where "Attribute Name" has the following format:</p> <p>Lxx_Ryy_Vzz, where:</p> <ul style="list-style-type: none"> • L means language • R means region • V means variant <p>Example:</p> <pre>UCA400R1_LSV (Swedish) "Kypper" < "Köpfe"</pre> <p>For long path equivalent setting, see CUNBOPRM_Locale description.</p> |
| Strength | S | 1, 2, 3, 4, I, D | <p>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).</p> <p>Example:</p> <pre>UCA400R1_S1 role = Role = rôle UCA400R1_S2 role = Role < rôle UCA400R1_S3 role < Role < rôle</pre> <p>For long path equivalent setting, see CUNBOPRM_Coll_Level description.</p> |

| Table 26. Collation keywords descriptions (continued) | | | |
|---|-----|-----------------|---|
| Attribute Name | Key | Possible Values | Description |
| Case_Level | K | X, O, D | <p>The Case Level attribute is used when ignoring accents but not case. In such case, set Strength to Primary, and Case_Level to On.</p> <p>In most locales, this setting is Off by default.</p> <p>Example:</p> <pre>UCA400R1_S1_KX role = Role = rôle UCA400R1_S1_KO role = rôle < Role</pre> <p>For long path equivalent setting, see CUNBOPRM_Case_Level description.</p> |
| Case_First | C | X, L, U, D | <p>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.</p> <p>Example:</p> <pre>UCA400R1_CX or UCA400R1_CL "china" < "China" < "denmark" < "Denmark" UCA400R1_CU "China" < "china" < "Denmark" < "denmark"</pre> <p>For long path equivalent setting, see CUNBOPRM_Case_First description.</p> |
| Alternate | A | N, S, D | <p>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.</p> <p>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.</p> <p>If Alternate is not set to Shifted, then there is no difference between a Strength of 3 and a Strength of 4.</p> <p>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.</p> <p>Example:</p> <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> <p>For long path equivalent setting, see CUNBOPRM_Variable_Opt description.</p> |
| Variable_Top | T | <hex digits> | <p>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.</p> <p>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".</p> <p>Example:</p> <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> <p>For long path equivalent setting, see CUNBOPRM_Var_Top description.</p> |

| Table 26. Collation keywords descriptions (continued) | | | |
|---|-----|-----------------|---|
| Attribute Name | Key | Possible Values | Description |
| Normalization Checking | N | X, O, D | <p>The Normalization setting determines whether text is thoroughly normalized or not in comparison (see also CUN4BOPR_Norm_Type).</p> <p>Example:</p> <pre>UCA400R1_NX ä= a + ì% < ä+ ì% < ;+ ì% UCA400R1_NO ä= a + ì% < ä+ ì% < ;+ ì%</pre> <p>For long path equivalent setting, see CUNBOPRM_Norm_Type description.</p> |
| French | F | X, O, D | <p>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).</p> <p>Example:</p> <pre>UCA400R1_FX cote < coté< côte < côté UCA400R1_F0 cote < côte< coté < côté</pre> <p>For long path equivalent setting, see CUNBOPRM_Cmp_Order description.</p> |
| Hiragana | H | X, O, D | <p>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.</p> <p>Example:</p> <pre>UCA400R1_HX_S4 M0... = -ã< M0†= -0æ UCA400R1_H0_S4 M0... < -ã< M0†< -0æ</pre> <p>For long path equivalent setting, see CUNBOPRM_Hiragana description.</p> |

Valid values for collation keywords are listed in the following table:

| Table 27. 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 one of the following Collation versions followed by desired sets:

- *UCA400R1_...
- *UCA410_...
- *UCA600_...
- *UCA900_...
- *UCAX13_...

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 28. 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 "&ae << æ" is equivalent to "&a << æ/e". |
| | a b | 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 29. 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 the example that follows table "Collation syntax rules". | 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. |
| Strength | [strength 1] [strength 2] [strength 3] [strength 4] [strength 5] [strength I] | 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. |

| Table 29. Collation syntax rules (continued) | | |
|--|--------------------------------------|---|
| Option | Example | Description |
| [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 in this topic.
- 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 (CUN4BOID).

```
* *****
* *                               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 30. 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)
xxx represents level two (two bytes)
yy  represents level three (one byte)
zzz 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:

```
02870858085B024102410241000000200020002000200020002000000020404041616
```

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 Collation versions (UCA400R1 and higher). Even the size of the sort key per Code Point might have many variations according the settings. For target buffers size, see section [“Target buffer length considerations”](#) on page 165.

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 31. 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 |

Table 31. Size of the work buffers for UTF-16BE Code Points (continued)

| Collation Level / Strength | Work Buffer length per Code Point in Source buffer |
|----------------------------|--|
| 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 [“Target buffer length considerations”](#) on page 165. 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 5, “Normalization,”](#) on page 97 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 32. 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 33. 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 |
| 2 | 7 - Bytes | 3 - Bytes |
| 3 | 9 - Bytes | 2 - Bytes |
| 4 | 12 - Bytes | 2 - Bytes |

Table 33. 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 |
|----------------------------|--|--|
| 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 34. 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 163 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 35. 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 | |
| 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 |

Table 35. 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 |
|--------------|--------------|--------------|----------------|----------------|-------------|-----------|------------|
| CUNSO06A | | X | | X | UCA400R1 | X | |
| CUNSO07A | | X | | X | UCA400R1 | | X |

Chapter 7. Bidi transformation

This topic describes the programming required for the bidi transformation service.

Note: IBM does not intend to enhance the bidi transformation service. Instead, it is recommended that you use the character conversion 'extended bidi support' for all new development and for the highest level of bidi support.

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 *Unicode Bidirectional Algorithm* (For z/OS v1R8 bidi only supports mirroring and character inversion). For further information about the bidi and character shaping service, see *Unicode Bidirectional Algorithm*, which is available in the [Unicode Technical Reports \(www.unicode.org/reports\)](http://www.unicode.org/reports).

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 technique B, which makes the transformation on the output buffer but preserving the current behavior.

Calling bidi transformation service

This topic 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” on page 170](#).

```
#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 */
        Bidi_ImpAlg : 1, /* 0 = Context LTR
                          /* 1 = Context RTL
                          /* Bidi Implicit Alg
                          /* 0 = Algor Basic
                          /* 1 = Algor Implicit
        : 6; /* FLAG Byte 1 set by caller*/
    } 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; /* FLAG Byte 1 set by caller*/
    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
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_CCSID_Src
LA    R15,CUNBBPRM_TRGCCSID Initialize String Type Trg
ST    R15,CUNBBPRM_CCSID_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 36. 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 |
| 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 |

Table 36. Mapping of parameters in HLASM for bidi AMODE (31) (continued)

| Offset Dec | Offset Hex | Type | Length in Bytes | Boundary | Name | Description |
|------------|------------|-----------|-----------------|----------|-----------------------|---|
| 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 |
| 88 | (58) | 1... .. | 1 | | CUNBBPRM_Bidi_Context | Bidi Context: 0=Context LTR 1=Context RTL |
| 88 | (58) | .1... .. | 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.

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 37. 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 |

Table 37. Mapping of parameters in HLASM for bidi AMODE (64) (continued)

| Offset Dec | Offset Hex | Type | Length in Bytes | Boundary | Name | Description |
|------------|------------|-----------|-----------------|----------|------------------------|---|
| 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 |
| 88 | (58) | 1... .. | 1 | | CUN4BBPR_Bidi_Context | Bidi Context: 0=Context LTR 1=Context RTL |
| 88 | (58) | .1... .. | 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 [“Calling the bidi conversion services” on page 17](#).

The 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.

Chapter 8. Stringprep conversion

This topic 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).

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 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 [RFC 3454 \(tools.ietf.org/html/rfc3454\)](https://tools.ietf.org/html/rfc3454).

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” on page 180](#).

```
#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];
```

Stringprep conversion

```
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             */
                                /* FLAG Byte 1 set by caller*/
        : 2;      /* 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      */
    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             */
                                /* FLAG Byte 1 set by caller*/
    } 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

```

Stringprep conversion

```
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 38. 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 |

Table 38. 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 |
|------------|------------|-----------|-----------------|----------|------------------------|--|
| 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 |
| 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 | DWORD | 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 |
| 140 | (8C) | 000. | 1 | | * | Reserved |
| 140 | (8C) | ...1 | 1 | | CUNBPPRM_UTF_Version | UTF version to use: 0000 = UTF-8 0001 = UTF-16 |

| Table 38. 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 |
| 140 | (8C) | 1... | 1 | | CUNBPPRM_UnassignedEr | If an unassigned code point found: 0 = Terminate processing and sets RC=8 1 = Continues processing |
| 140 | (8C) |1.. | 1 | | CUNBPPRM_Page_fix | Page fix: 0 = No Page fix 1 = Page fix |
| 140 | (8C) |11 | 1 | | * | 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.

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 stringprep is to be terminated with an error.
- **1:** Indicates that the stringprep is to be given a warning and continues processing.

CUNBPPRM_Page_Fix

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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 39. 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 |

| Table 39. 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 |
| 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 | DWORD | 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 |
| 136 | (88) | BITSTRING | 1 | | CUN4BPPR_Flags | Flags |
| 136 | (88) | 000. | 1 | | * | Reserved |
| 136 | (88) | ...1 | 1 | | CUN4BPPR_UTF_Version | UTF version to use: 0000 = UTF-8 0001 = UTF-16 |
| 136 | (88) | 1... | 1 | | CUN4BPPR_UnassignedEr | If an unassigned code point found: 0 = Terminate processing and sets RC=8 1 = Continues processing |
| 136 | (88) |1.. | 1 | | CUN4BPPR_Page_fix | Page fix: 0 = No Page fix 1 = Page fix |
| 136 | (88) |11 | 1 | | * | 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.

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 stringprep is to be terminated with an error.
- **1:** Indicates that the stringprep is to be given a warning and continues processing.

CUN4BPPR_Page_Fix

If the requested conversion is not currently loaded in memory, this flag indicates if it should be loaded in page-fixed memory.

- **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

Chapter 9. Conversion information service

This topic 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 **CUN4LINF** 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.

Note: The information returned by this service reflects the status when the release was made available.

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 40. 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 |
| 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” on page 192. 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              */
                                           /* CCSID1 Info -----              */
    unsigned int    CCSID1;                 /* CCSID1                          */
    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 */
    char Res3[4];                              /* Reserved */
    } CCSID1_Num_Subs;                         /* 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 */
    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 */
    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 */
        /* Reserved */
        /* Out Flags-Set by the Service */
    } Gen_Flags_Out;
    struct {
        int Get_Tech_Supp_fCCSID2_tCCSID1 : 1, /* Get techniques supported from
        /* CCSID2 to CCSID1 */
        /* 0 - Do not obtain techniques
        /* from CCSID2 to CCSID1
        /* (default) */
        /* 1 - Obtain techniques
        /* from CCSID2 to CCSID1
        /* Reserved */
        /* In Flags-Set by the Caller */
        /* Reserved */
        /* Conversion techniques sup-
        /* ported From CCSID1 To
        /* CCSID2
        /* Note. Meaningful in case that
        /* Conversion_Supported is
        /* Turned ON
    } Gen_Flags_In;
    char Res7[6];                /* Reserved */
    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];                /* Reserved */
    void * DDA_Buf_Ptr;           /* Dynamic data area pointer */
    unsigned int DDA_Buf_ALET;     /* Dynamic data area ALET */
    unsigned int DDA_Buf_Len;      /* Dynamic data area length */
    /* RC / RS */
    struct {
        int Return_Code;          /* Return_Code */
        int Reason_Code;          /* Reason_Code */
    } RC_RS;                     /* Return/Reason code */

    /******
    /* Additional information for Version 2 Parameter Area */
    /******
    char CCSID1_SUFFIX[2];        /* Suffix for CCSID1. The suffix
    /* for subCCSIDs are returned in
    /* subCCSIDs_info
    char CCSID2_SUFFIX[2];        /* Suffix for CCSID1. The suffix
    /* for subCCSIDs are returned in
    /* subCCSIDs_info
    unsigned char Conversion_Type; /* type of conversion for
    /* CCSID1 to CCSID2
    /* 1 = direct conversion
    /* 2 = indirect conversion
    char Resi1[3];                /* Reserved */

```

```

void *      CCSID1_CTLDEF_Ptr ;      /* Pointer to CTLF (optional)      */
unsigned int CCSID1_CTLDEF_ALET;     /* ALET for CTLF_Ptr               */
unsigned char CCSID1_CTLDEF_Num ;    /* Num of entries in CTLF          */
char Resi2[3];                      /* Reserved                         */

void *      CCSID2_CTLDEF_Ptr ;      /* Pointer to CTLF                  */
unsigned int CCSID2_CTLDEF_ALET;     /* ALET for CTLF_Ptr               */
unsigned char CCSID2_CTLDEF_Num ;    /* Num of entries in CTLF          */
char Resi3[11];                     /* Reserved                         */
} CUNBIPRM;

typedef struct tagCUNBIPRM_CTLF {
    unsigned int  CF_CCSID;           /* Control Function Definitions     */
    char          resCF1[4];          /* CCSID or subCCSID               */
    struct {
        unsigned char  CF_SP_State;  /* reserved                         */
        unsigned char  CF_SP_Num;    /* CF_DEFS                         */
        short int      CF_SP_Width; /* space character state           */
        struct {
            char        CF_SP_2[2];  /* number of space character       */
            struct {
                char     CF_SP_S2[1]; /* space character width           */
                char     CF_SP_S1[1]; /* space character code point      */
            } CF_SP_1;              /* UCS-2: single wide space char  */
        } CF_SP_Code;              /* UCS-2: double wide space char */
        unsigned char  CF_SUB_State; /* second single wide space char  */
        unsigned char  CF_SUB_Num;  /* first single wide space char   */
        short int      CF_SUB_Width; /* first double wide space char   */
        struct {
            char        CF_SUB_2[2]; /* sub character state            */
            struct {
                char     CF_SUB_S2[1]; /* number of sub character        */
                char     CF_SUB_S1[1]; /* sub character width            */
            } CF_SUB_1;              /* sub character code point       */
        } CF_SUB_Code;              /* UCS-2: single wide sub char   */
        unsigned char  CF_NL_State;  /* UCS-2: double wide sub char   */
        char          resNL[1];      /* second single wide sub char    */
        short int      CF_NL_Width;  /* first single wide sub char     */
        char          CF_NL_Code[4]; /* first double wide sub char     */
        unsigned char  CF_LF_State;  /* sub character state            */
        char          resLF[1];      /* number of sub character        */
        short int      CF_LF_Width;  /* sub character width            */
        char          CF_LF_Code[4]; /* sub character code point       */
        unsigned char  CF_CR_State;  /* UCS-2: single wide sub char   */
        char          resCR[1];      /* UCS-2: double wide sub char   */
        short int      CF_CR_Width;  /* second single wide sub char    */
        char          CF_CR_Code[4]; /* first single wide sub char     */
        unsigned char  CF_EOF_State; /* first double wide sub char    */
        char          resEOF[1];     /* sub character state            */
        short int      CF_EOF_Width; /* number of sub character        */
        char          CF_EOF_Code[4]; /* sub character width            */
    } CF_DEFS;
    char          resCF2[8];          /* reserved                         */
} CUNBIPRM_CTLF;

```

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

64-bit mapping

```

typedef struct tagCUN4BIPR {
    unsigned int  Version;           /* Structure version number        */
    unsigned int  Length;            /* Length of structure             */
    unsigned int  CCSID1;            /* CCSID1 Info -----            */
    struct {
        char Res1[2];               /* CCSID1                          */
        short int CCSID1_ES_ID;      /* Reserved                         */
        char CCSID1_ES_Name[28];     /* Encoding Scheme ID              */
    } CCSID1_ES;                   /* Encoding Scheme Name            */
    struct {
        unsigned char CCSID1_ES_Size_Min; /* CCSID1 Encoding Scheme info */
        unsigned char CCSID1_ES_Size_Max; /* ES Size Min                    */
    } CCSID1_ES_Size;              /* ES Size Max                    */
} CUN4BIPR;

```

```

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 */
    char Res3[4]; /* Reserved */
} CCSID1_Num_Subs; /* 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 */
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 */
    : 5; /* Reserved */
} Gen_Flags_Out; /* Out Flags-Set by the Service */
struct {
    int Get_Tech_Supp_fCCSID2_tCCSID1 : 1, /* Get techniques supported from */
    /* CCSID2 to CCSID1 */
    /* 0 - Do not obtain techniques */
    /* from CCSID2 to CCSID1 */
    /* (default) */
    : 7; /* 1 - Obtain techniques */
    /* from CCSID2 to CCSID1 */
    /* Reserved */
} Gen_Flags_In; /* In Flags-Set by the Caller */
char Res6[6]; /* Reserved */
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 ----- */
void * DDA_Buf_Ptr; /* Dynamic data area pointer */
unsigned int DDA_Buf_ALET; /* Dynamic data area ALET */
unsigned int DDA_Buf_Len; /* Dynamic data area length */
/* RC / RS */
struct {
    int Return_Code; /* Return_Code */
    int Reason_Code; /* Reason_Code */
} RC_RS; /* Return/Reason code */

/*****
/* Additional information for Version 2 Parameter Area */
/*****
char CCSID1_SUFFIX[2]; /* Suffix for CCSID1. The suffix */
/* for subCCSIDs are returned in */
/* subCCSIDs_info */
char CCSID2_SUFFIX[2]; /* Suffix for CCSID1. The suffix */
/* for subCCSIDs are returned in */
/* subCCSIDs_info */
unsigned char Conversion_Type; /* type of conversion for */
/* CCSID1 to CCSID2 */
/* 1 = direct conversion */
/* 2 = indirect conversion */
char Resi1[3]; /* Reserved */

void * CCSID1_CTLDEF_Ptr ; /* Pointer to CTLF (optional) */
unsigned int CCSID1_CTLDEF_ALET; /* ALET for CTLF_Ptr */
unsigned char CCSID1_CTLDEF_Num ; /* Num of entries in CTLF */
char Resi2[3]; /* Reserved */

void * CCSID2_CTLDEF_Ptr ; /* Pointer to CTLF */
unsigned int CCSID2_CTLDEF_ALET; /* ALET for CTLF_Ptr */
unsigned char CCSID2_CTLDEF_Num ; /* Num of entries in CTLF */

```

```

char Resi3[11];                /* Reserved */
} CUN4BIPR;

typedef struct tagCUN4BIPR_CTLF {
    unsigned int  CF_CCSID;      /* Control Function Definitions */
    char          resCF1[4];     /* CCSID or subCCSID */
    struct {                /* reserved */
        unsigned char  CF_SP_State; /* CF_DEFS */
        unsigned char  CF_SP_Num;   /* space character state */
        short int      CF_SP_Width; /* number of space character */
        struct {        /* space character width */
            char      CF_SP_2[2];    /* space character code point */
            struct {
                char      CF_SP_S2[1]; /* UCS-2: single wide space char */
                char      CF_SP_S1[1]; /* UCS-2: double wide space char */
            } CF_SP_1; /* second single wide space char */
        } CF_SP_Code; /* first single wide space char */
        /* first double wide space char */

        unsigned char  CF_SUB_State; /* sub character state */
        unsigned char  CF_SUB_Num;   /* number of sub character */
        short int      CF_SUB_Width; /* sub character width */
        struct {        /* sub character code point */
            char      CF_SUB_2[2];    /* UCS-2: single wide sub char */
            struct {
                char      CF_SUB_S2[1]; /* UCS-2: double wide sub char */
                char      CF_SUB_S1[1]; /* second single wide sub char */
            } CF_SUB_1; /* first single wide sub char */
        } CF_SUB_Code; /* first double wide sub char */

        unsigned char  CF_NL_State; /* New Line character state */
        char          resNL[1];     /* reserved */
        short int      CF_NL_Width; /* New Line character width */
        char          CF_NL_Code[4]; /* New Line character Code point */
        unsigned char  CF_LF_State; /* Line Feed character state */
        char          resLF[1];     /* reserved */
        short int      CF_LF_Width; /* Line Feed character width */
        char          CF_LF_Code[4]; /* Line Feed character Code point */
        unsigned char  CF_CR_State; /* Carriage Return state */
        char          resCR[1];     /* reserved */
        short int      CF_CR_Width; /* Carriage Return width */
        char          CF_CR_Code[4]; /* Carriage Return code point */
        unsigned char  CF_EOF_State; /* End-Of-File character state */
        char          resEOF[1];    /* reserved */
        short int      CF_EOF_Width; /* EOF character width */
        char          CF_EOF_Code[4]; /* EOF character Code point */
    } CF_DEFS;

    char          resCF2[8];        /* reserved */
} CUN4BIPR_CTLF;

```

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, CUNSISMA, is provided in SYS1.SAMPLIB.

```

For AMODE (31)
-----1-----2-----3-----4-----5-----6-----7--
      EJECT
CUNSISMA CSECT
CUNSISMA AMODE 31
CUNSISMA RMODE ANY
      SPACE 1
PSTART BRAS R15,PSTART      ! ESTABLISH ADDRESSABILITY
      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                 ! Cleaning...
*
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
SPACE 1                 ! 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
*****
*
* DECLARATION
*
*
* Section
*
*
*****
*****
* 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
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
PSTART   BRAS    R15,PSTART              ! ESTABLISH ADDRESSABILITY
          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          ! CLEAR PARAMETER AREA
LHI R3,CUN4BIPR_LEN      ! PA Address
LHI R15,0                 ! PA Len
MVCL R2,R14              ! Filler - Nulls
                           ! Cleaning...
SPACE 1
*
LA R0,CUN4BIPR_VER        ! SETTING PA VERSION
ST R0,CUN4BIPR_VERSION    ! GET ACTUAL VERSION
LA R0,CUN4BIPR_LEN        ! STORE INTO PARAMETER
ST R0,CUN4BIPR_LENGTH     ! GET ACTUAL 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_ALËT,=F'0'
L R0,=A(CUN4BIPR_DDA_REQ)
ST R0,CUN4BIPR_DDA_BUF_LEN
SPACE 1
*****
*          CALLING THE CNV INFO SERVICE          *
*****
SPACE 1
CALL CUN4LINP,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
*****
*          DECLARATION          *
*
*          Section          *
*
*****
*****
*          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'
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 41. 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_CCSID1 | Specify CCSID1 |
| 12 | (C) | CHARACTER | 32 | WORD | CUNBIPRM_CCSID1_ES | CCSID1 encoding scheme (ES) information |
| 12 | (C) | CHARACTER | 2 | | * | Reserved |
| 14 | (E) | UNSIGNED | 2 | | CUNBIPRM_CCSID1_ES_ID | Encoding scheme ID for CCSID1 |
| 16 | (10) | CHARACTER | 28 | | CUNBIPRM_CCSID1_ES_Name | Encoding scheme name for CCSID1 |
| 44 | (2C) | CHARACTER | 2 | | CUNBIPRM_CCSID1_ES_Size | Encoding scheme size for CCSID1 |
| 44 | (2C) | UNSIGNED | 1 | | CUNBIPRM_CCSID1_ES_Size_Min | Minimum encoding scheme size for CCSID1 |

Table 41. 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 |
|------------|------------|-----------|-----------------|----------|---------------------------------|--|
| 45 | (2D) | UNSIGNED | 1 | | CUNBIPRM_CCSID1_ES_Size_Max | Maximum encoding scheme size for CCSID1 |
| 46 | (2E) | CHARACTER | 2 | | * | Reserved |
| 48 | (30) | CHARACTER | 64 | | CUNBIPRM_CCSID1_Description | CCSID1 description |
| 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 |

Table 41. 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 |
|------------|------------|-----------|-----------------|----------|-------------------------------------|--|
| 132 | (84) | CHARACTER | 4 | | CUNBIPRM_CCSID1_Sub_Char_QBCS_1 | The second substitution character for the QBCS |
| 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 41. 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_Subs_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 |
| 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 |

Table 41. 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 |
|------------|------------|-----------|-----------------|----------|---|---|
| 312 | (138) | BITSTRING | 1 | | CUNBIPRM_Gen_Flags _Out | Out-FLAG Byte 1 (Set by the service) |
| 312 | (138) | 1... .. | 1 | | CUNBIPRM_CCSID1 _Supported | CCSID1 supported: 0=CCSID1 is not supported. 1=CCSID1 is supported Meaningful if only CCSID1 is provided. |
| 312 | (138) | .1... .. | 1 | | CUNBIPRM_CCSID2 _Supported | CCSID2 supported: 0=CCSID2 is not supported. 1=CCSID2 is supported. Meaningful if only CCSID2 is provided. |
| 312 | (138) | ..1. | 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. |
| 336 | (150) | CHARACTER | 4 | | * | Reserved |
| 340 | (154) | ADDRESS | 4 | DWORD | CUNBIPRM_DDA_Buf _Ptr | Dynamic data area pointer |
| 344 | (158) | UNSIGNED | 4 | | CUNBIPRM_DDA_Buf _ALET | Dynamic data area ALET |

Table 41. 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 |
|------------|------------|-----------|-----------------|----------|-----------------------------|---|
| 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 | 2 | | CUNBIPRM_CCSID1_SUFFIX | Suffix for CCSID1 |
| 362 | (16A) | CHARACTER | 2 | | CUNBIPRM_CCSID2_SUFFIX | Suffix for CCSID2 |
| 364 | (16C) | UNSIGNED | 1 | | CUNBIPRM_Conversion_Type | Type of conversion for CCSID1 to CCSID2 1 = direct conversion 2 = indirect conversion |
| 365 | (16D) | CHARACTER | 3 | | * | Reserved |
| 368 | (170) | ADDRESS | 4 | | CUNBIPRM_CCSID1_CTLDEF_Ptr | Optional pointer to CCSID1 CUNBIPRM_CTLF |
| 372 | (174) | UNSIGNED | 4 | | CUNBIPRM_CCSID1_CTLDEF_Alet | ALET for CUNBIPRM_CCSID1_CTLDEF_Ptr |
| 376 | (178) | UNSIGNED | 1 | | CUNBIPRM_CCSID1_CTLDEF_Num | Number of entries |
| 377 | (179) | CHARACTER | 3 | | * | Reserved |
| 380 | (17C) | ADDRESS | 4 | | CUNBIPRM_CCSID2_CTLDEF_Ptr | Optional pointer to CCSID2 CUNBIPRM_CTLF |
| 384 | (180) | UNSIGNED | 4 | | CUNBIPRM_CCSID2_CTLDEF_Alet | ALET for CUNBIPRM_CCSID2_CTLDEF_Ptr |
| 388 | (184) | UNSIGNED | 1 | | CUNBIPRM_CCSID2_CTLDEF_Num | Number of entries |
| 389 | (185) | CHARACTER | 11 | | * | Reserved |
| 400 | (190) | 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

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

CUNBIPRM_Ver

Supported on z/OS V1R10 and later releases.

CUNBIPRM_Ver2

Supported on z/OS V1R13 and later releases. This version provides additional CCSID information:

- CCSID suffixes
- Control character definitions
- Conversion type (direct or indirect)

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 the conversion is between CCSID 1200 and an unmixed CCSID 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.

See [Table 58 on page 309](#) for the ES IDs and the ES names table.

For more information about encoding schemes, see [“Encoding Scheme” on page 309](#).

CUNBIPRM_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), CUNBIPRM_CCSID1_ES_Size_Min and CUNBIPRM_CCSID1_ES_Size_Max contain different values; otherwise, they contain the same value.

CUNBIPRM_CCSID1_ES_Size_Min - set by the service

Specifies the minimum character set byte size for CCSID1.

CUNBIPRM_CCSID1_ES_Size_Max - set by the service

Specifies the maximum character set byte size for CCSID1.

CUNBIPRM_CCSID1_Description - set by the service

Specifies the description of CCSID1 (data returned encoded in CCSID 37).

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_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 CUNBIPRM_CCSID2_ are not meaningful after the service is called.

This field is updated by the service when the conversion is between CCSID 1200 and an unmixed CCSID, returning the latest Unicode versions available for conversion between CCSID1 and CCSID2. z/OS Unicode conversion information service updates this field only when both CCSIDs are provided. For individual CCSID requests, CUNBIPRM_CCSID2 remains unchanged even when CCSID 1200 is specified.

CUNBIPRM_CCSID2_ES - set by the service

Specifies the ES information in the following fields:

CUNBIPRM_CCSID2_ES_ID - set by the service

Specifies the ES ID for the specified CCSID2.

CUNBIPRM_CCSID1_ES_Name - set by the service

Specifies the ES name for the specified CCSID2.

See [Table 58 on page 309](#) for the ES IDs and the ES names table.

For more information about encoding schemes, see [“Encoding Scheme” on page 309](#).

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 (returned encoded in CCSID 37).

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 to 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_Conversion_Type - set by the service

Specifies the type of conversion from CCSID1 to CCSID2, according to the following values:

1

Direct conversion.

2

Indirect conversion.

CUNBIPRM_Conversion_Type is meaningful when the CUNBIPRM_Conversion_Supported bit is on.

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 to 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_CCSID1_SUFFIX - set by the service

Specifies the suffix for CCSID1. For a mixed CCSID, the suffix for subCCSIDs are returned in CUNBIPRM_subCCSIDs_info.

CUNBIPRM_CCSID2_SUFFIX - set by the service

Specifies the suffix for CCSID2. For a mixed CCSID, the suffix for subCCSIDs are returned in CUNBIPRM_subCCSIDs_info.

CUNBIPRM_CCSID1_CTLDEF_Ptr (optional) - set by caller

For parameter area version 2, specifies an optional buffer where z/OS Unicode conversion service information service retrieves information for all the control character definitions for CCSID1.

If CCSID1 is a mixed CCSID, the buffer will contain the control character definitions for all the subCCSIDs. IBM recommends that when CUNBIPRM_CCSID1_CTLDEF_Ptr is specified, verify the contents of CUNBIPRM_CCSID1_CTLDEF_Num after the service is called successfully. If CUNBIPRM_CCSID1_CTLDEF_Num > 1, CCSID1 is a mixed CCSID.

CUNBIPRM_CTLF can be addressed by CUNBIPRM_CCSID1_CTLDEF_Ptr and CUNBIPRM_CCSID1_CTLDEF_ALET making a loop CUNBIPRM_CCSID1_CTLDEF_Num times by the length of CUNBIPRM_CTLF in order to obtain information for the different subCCSIDs that belong to mixed CCSID1.

The size of this buffer must be allocated according to the content of version 2 CUNBIPRM_CTLF_Len_Req in a double-word boundary area. CUNBIPRM_CTLF_Len_Req is provided in the IDF file CUNBIIDF.

CUNBIPRM_CCSID1_CTLDEF_ALET- set by caller

Specifies the ALET for CUNBIPRM_CCSID1_CTLDEF_Ptr and is required only if CUNBIPRM_CCSID1_CTLDEF_Ptr is specified.

CUNBIPRM_CCSID1_CTLDEF_Num - set by the service

Specifies the number of entries in CUNBIPRM_CCSID1_CTLF buffer. If CUNBIPRM_CCSID1_CTLDEF_Num is equal to 1, CCSID1 is not a mixed conversion. If CUNBIPRM_CCSID1_CTLDEF_Num is > 1, CCSID1 is a mixed CCSID.

CUNBIPRM_CCSID2_CTLDEF_Ptr (optional) - set by caller

For parameter area version 2, specifies an optional buffer where z/OS Unicode conversion service information service retrieves information for all the control character definitions for CCSID2.

If CCSID2 is a mixed CCSID, the buffer will contain the control character definitions for all the subCCSIDs. IBM recommends that when CUNBIPRM_CCSID2_CTLDEF_Ptr is specified, verify the contents of CUNBIPRM_CCSID2_CTLDEF_Num after the service is called successfully. If CUNBIPRM_CCSID2_CTLDEF_Num > 1, CCSID2 is a mixed CCSID.

CUNBIPRM_CTLF can be addressed by CUNBIPRM_CCSID2_CTLDEF_Ptr and CUNBIPRM_CCSID2_CTLDEF_ALET making a loop CUNBIPRM_CCSID2_CTLDEF_Num times by the length of CUNBIPRM_CTLF in order to obtain information for the different subCCSIDs that belong to mixed CCSID2.

The size of this buffer must be allocated according to the content of version 2 CUNBIPRM_CTLF_Len_Req in a double-word boundary area. CUNBIPRM_CTLF_Len_Req is provided in the IDF file CUNBIIDF.

CUNBIPRM_CCSID2_CTLDEF_ALET- set by caller

Specifies the ALET for CUNBIPRM_CCSID2_CTLDEF_Ptr and is required only if CUNBIPRM_CCSID2_CTLDEF_Ptr is specified.

CUNBIPRM_CCSID2_CTLDEF_Num - set by the service

Specifies the number of entries in CUNBIPRM_CCSID2_CTLF buffer. If CUNBIPRM_CCSID2_CTLDEF_Num is equal to 1, CCSID2 is not a mixed conversion. If CUNBIPRM_CCSID2_CTLDEF_Num is > 1, CCSID2 is a mixed CCSID.

CUNBIPRM_CTLF - set by the service

CUNBIPRM_CF_CCSID - set by the service

If the input CCSID is a mixed CCSID, this specifies the sub CCSID. Otherwise, this specifies the input CCSID.

CUNBIPRM_CF_SP_STATE - set by the service

The state number of the space character in which the code point is to be used.

CUNBIPRM_CF_SP_NUM - set by the service

The number of space characters in this element for this CCSID.

CUNBIPRM_CF_SP_WIDTH - set by the service

The width of the space character code point.

CUNBIPRM_CF_SP_CODE - set by the service

The space character code point.

For UCS-2, the four bytes code point are divided in two halves. The left most two bytes are for single wide space and the right most two bytes are for double wide space.

For UTF-32, the actual length for space character is 4-bytes. However, the code point values returned are two bytes for each space character so left paddings with zeros to four bytes are needed before use. For example, for UTF-32, a value of x'0020' should be padded to 4-bytes as x'00000020' before use.

CUNBIPRM_CF_SUB_STATE - set by the service

The state number of the sub character in which the code point is to be used.

CUNBIPRM_CF_SUB_NUM - set by the service

The number of sub characters in this element for this CCSID.

CUNBIPRM_CF_SUB_WIDTH - set by the service

The width of the sub character code point.

CUNBIPRM_CF_SUB_CODE - set by the service

The substitution character code point.

For UCS-2, the four bytes are divided in two halves. The left most two bytes are used for conversions from SBCS, and the right most two bytes are for conversions from MBCS.

For UTF-32, the actual length for substitution character is 4-bytes. However, the code point values returned are two bytes for each substitution character so left paddings with zeros to four bytes are needed before use. For example, for UTF-32, a value of x'001A' and x'FFFD' should be padded to x'0000001A' and x'0000FFFF' before use.

CUNBIPRM_CF_NL_STATE - set by the service

The state number of the new line character in which the code point is to be used.

CUNBIPRM_CF_NL_WIDTH - set by the service

The width of the new line character code point.

CUNBIPRM_CF_NL_CODE - set by the service

The new line character code point.

There is only one new line character code point for each CCSID or sub CCSID, and the code point is right aligned.

CUNBIPRM_CF_LF_STATE - set by the service

The state number of the line feed character in which the code point is to be used.

CUNBIPRM_CF_LF_WIDTH - set by the service

The width of the line feed character code point.

CUNBIPRM_CF_LF_CODE - set by the service

The line feed character code point.

There is only one line feed character code point for each CCSID or sub CCSID, and the code point is right aligned.

CUNBIPRM_CF_CR_STATE - set by the service

The state number of the carriage return control character in which the code point is to be used.

CUNBIPRM_CF_CR_WIDTH - set by the service

The width of the carriage return control character code point.

CUNBIPRM_CF_CR_CODE - set by the service

The carriage return character code point.

There is only one carriage return character code point for each CCSID or sub CCSID, and the code point is right aligned.

CUNBIPRM_CF_EOF_STATE - set by the service

The state number of the end of file character in which the code point is to be used.

CUNBIPRM_CF_EOF_WIDTH - set by the service

The width of the end of file character code point.

CUNBIPRM_CF_EOF_CODE - set by the service

The end of file character code point.

There is only one end of file character code point for each CCSID or sub CCSID, and the code point is right aligned.

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_Info - set by the service**CUNBIPRM_subCCSIDs_CCSID - set by the service**

Specifies subCCSIDs.

CUNBIPRM_subCCSIDs_Size - set by the service

Specifies the size character for the subCCSID.

CUNBIPRM_subCCSIDs_Suffix - set by the service

Specifies the suffix characters 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.

| Table 42. 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 |

Table 42. 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 |
|------------|------------|-----------|-----------------|----------|-------------------------------------|--|
| 120 | (78) | CHARACTER | 1 | | CUN4BIPR_CCSID1_Sub_Char_SBCS_1 | The second substitution character for the SBCS |
| 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 |
| 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 |

Table 42. 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 |
|-----------------------|-----------------------|-------------|----------------------------|-----------------|-------------------------------------|---|
| 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 |
| 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 |

Table 42. 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 |
|------------|------------|-----------|-----------------|----------|-------------------------------------|---|
| 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 |
| 312 | (138) | BITSTRING | 1 | | CUN4BIPR_Gen_Flags_Out | Out-FLAG Byte 1 (Set by the service) |
| 312 | (138) | 1... .. | 1 | | CUN4BIPR_CCSID1_Supported | CCSID1 supported: 0=CCSID1 is not supported. 1=CCSID1 is supported. Meaningful if only CCSID1 is provided. |
| 312 | (138) | .1... .. | 1 | | CUN4BIPR_CCSID2_Supported | CCSID2 supported: 0=CCSID2 is not supported. 1=CCSID2 is supported. Meaningful if only CCSID2 is provided. |
| 312 | (138) | ..1. | 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) |

Table 42. 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 |
|------------|------------|-----------|-----------------|----------|---|---|
| | | 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 | DWORD | 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 |
| 356 | (164) | UNSIGNED | 4 | | CUN4BIPR_Reason_Code | Reason code |
| 360 | (168) | CHARACTER | 2 | | CUN4BIPR_CCSID1_SUFFIX | Suffix for CCSID1 |
| 362 | (16A) | CHARACTER | 2 | | CUN4BIPR_CCSID2_SUFFIX | Suffix for CCSID2 |
| 364 | (16C) | UNSIGNED | 1 | | CUN4BIPR_Conversion_Type | Type of conversion for CCSID1 to CCSID2 1 = direct conversion 2 = indirect conversion |
| 365 | (16D) | CHARACTER | 3 | | * | Reserved |
| 368 | (170) | ADDRESS | 8 | | CUN4BIPR_CCSID1_CTLDEF_Ptr | Optional pointer to CCSID1 CUN4BIPR_CTLF |
| 376 | (178) | UNSIGNED | 4 | | CUN4BIPR_CCSID1_CTLDEF_Alet | ALET for CUN4BIPR_CCSID1_CTLDEF_Ptr |
| 380 | (17C) | UNSIGNED | 1 | | CUN4BIPR_CCSID1_CTLDEF_Num | Number of entries |
| 381 | (17D) | CHARACTER | 3 | | * | Reserved |
| 384 | (180) | ADDRESS | 8 | | CUN4BIPR_CCSID2_CTLDEF_Ptr | Optional pointer to CCSID2 CUN4BIPR_CTLF |

Table 42. 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 |
|------------|------------|-----------|-----------------|----------|-----------------------------|--|
| 392 | (188) | UNSIGNED | 4 | | CUN4BIPR_CCSID2_CTLDEF_Alet | ALET for CUN4BIPR_CCSID2_CTLDEF_Ptr |
| 396 | (18C) | UNSIGNED | 1 | | CUN4BIPR_CCSID2_CTLDEF_Num | Number of entries |
| 397 | (18D) | CHARACTER | 11 | | * | Reserved |
| 408 | (198) | 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

Specifies the version of the parameter area. This field must be initialized for the first call to stub routine CUNLININFO using the constant CUN4BIPR_Ver that is supplied by the interface definition file CUNBIIDF.

CUN4BIPR_Ver

Supported on z/OS V1R10 and later releases.

CUN4BIPR_Ver2

Supported on z/OS V1R13 and later releases. This version provides additional CCSID information:

- CCSID suffixes
- Control character definitions
- Conversion type (direct or indirect)

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_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 CUN4BIPR_CCSID1_ are not meaningful after calling the service.

This field is updated by the service when the conversion is between CCSID 1200 and an unmixed CCSID, 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_CCSID1* remains unchanged even when CCSID 1200 is specified.

CUN4BIPR_CCSID1_ES - set by the service

Specifies the encoding scheme (ES) information in the following fields:

CUN4BIPR_CCSID1_ES_ID - set by the service

Specifies the encoding scheme ID for the specified CCSID1.

CUN4BIPR_CCSID1_ES_Name - set by the service

Specifies the encoding scheme name for the specified CCSID1.

See Table 58 on page 309 for the ES IDs and the ES names table.

For more information about encoding schemes, see [“Encoding Scheme” on page 309](#).

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_Subs_DBCS - set by the service

Specifies the number of substitution characters to the DBCS that are involved by CCSID1.

CUN4BIPR_CCSID1_Num_Subs_TBCS - set by the service

Specifies the number of substitution characters to the TBCS that are involved by CCSID1.

CUN4BIPR_CCSID1_Num_Subs_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_Supprtd = 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_Subs_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_Subs_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_Subs_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_Subs_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_Subs_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_Subs_TBCS is equal to 1 or 2.

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_Subcs_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_Subcs_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 the conversion is between CCSID 1200 and an unmixed CCSID, 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.

See [Table 58 on page 309](#) for the ES IDs and the ES names table.

For more information about encoding schemes, see [“Encoding Scheme” on page 309](#).

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_Sub_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_Sub_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.

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_CCSID2_subCCSIDs_Info_ALET- set by caller

Specifies the alet for CUN4BIPR_CCSID2_subCCSIDs_Info_Ptr and is required if CUN4BIPR_CCSID2_subCCSIDs_Info_Ptr is specified only.

CUN4BIPR_CCSID2_subCCSIDs_Info_Num - set by the service

Specifies the number of subCCSIDs that belong to CCSID2. If CUN4BIPR_CCSID1_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_CCSID1_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_CCSID1_Supported is meaningful when CCSID1 is provided.

CUNBIPR_CCSID2_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_CCSID2_Supported is meaningful when CCSID2 is provided.

CUN4BIPR_Conversion_Supported - set by the service

Specifies whether the conversion between CCSIDs provided by CUN4BIPR_CCSID1 and CUN4BIPR_CCSID2 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_Conversion_Type - set by the service

Specifies the type of conversion from CCSID1 to CCSID2, according to the following values:

1

Direct conversion.

2

Indirect conversion.

CUN4BIPR_Conversion_Type is meaningful when the CUN4BIPR_Conversion_Supported bit is on

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_CCSID1_SUFFIX - set by the service

Specifies the suffix for CCSID1. For a mixed CCSID, the suffix for subCCSIDs are returned in CUN4BIPR_subCCSIDs_info.

CUN4BIPR_CCSID2_SUFFIX - set by the service

Specifies the suffix for CCSID2. For a mixed CCSID, the suffix for subCCSIDs are returned in CUN4BIPR_subCCSIDs_info.

CUN4BIPR_CCSID1_CTLDEF_Ptr (optional) - set by caller

For parameter area version 2, specifies an optional buffer where z/OS Unicode conversion service information service retrieves information for all the control character definitions for CCSID1. If CCSID1 is a mixed CCSID, the buffer will contain the control character definitions for all the subCCSIDs. IBM recommends that when CUN4BIPR_CCSID1_CTLDEF_Ptr is specified, verify the contents of CUN4BIPR_CCSID1_CTLDEF_Num after the service is called successfully. If CUN4BIPR_CCSID1_CTLDEF_Num > 1, CCSID1 is a mixed CCSID.

CUN4BIPR_CTLF can be addressed by CUN4BIPR_CCSID1_CTLDEF_Ptr and CUN4BIPR_CCSID1_CTLDEF_ALET making a loop CUN4BIPR_CCSID1_CTLDEF_Num times by the length of CUN4BIPR_CTLF in order to obtain information for the different subCCSIDs that belong to mixed CCSID1.

The size of this buffer must be allocated according to the content of version 2 CUN4BIPR_CTLF_Len_Req in a double-word boundary area. CUN4BIPR_CTLF_Len_Req is provided in the IDF file CUNBIIDF.

CUN4BIPR_CCSID1_CTLDEF_ALET- set by caller

Specifies the ALET for CUN4BIPR_CCSID1_CTLDEF_Ptr and is required only if CUN4BIPR_CCSID1_CTLDEF_Ptr is specified.

CUN4BIPR_CCSID1_CTLDEF_Num - set by the service

Specifies the number of entries in CUN4BIPR_CCSID1_CTLF buffer. If CUN4BIPR_CCSID1_CTLDEF_Num is equal to 1, CCSID1 is not a mixed conversion. If CUN4BIPR_CCSID1_CTLDEF_Num is > 1, CCSID1 is a mixed CCSID.

CUN4BIPR_CCSID2_CTLDEF_Ptr (optional) - set by caller

For parameter area version 2, specifies an optional buffer where z/OS Unicode conversion service information service retrieves information for all the control character definitions for

CCSID2. If CCSID2 is a mixed CCSID, the buffer will contain the control character definitions for all the subCCSIDs. IBM recommends that when CUN4BIPR_CCSID2_CTLDEF_Ptr is specified, verify the contents of CUN4BIPR_CCSID2_CTLDEF_Num after the service is called successfully. If CUN4BIPR_CCSID2_CTLDEF_Num > 1, CCSID2 is a mixed CCSID.

CUN4BIPR_CTLF can be addressed by CUN4BIPR_CCSID2_CTLDEF_Ptr and CUN4BIPR_CCSID2_CTLDEF_ALET making a loop CUN4BIPR_CCSID2_CTLDEF_Num times by the length of CUN4BIPR_CTLF in order to obtain information for the different subCCSIDs that belong to mixed CCSID2.

The size of this buffer must be allocated according to the content of version 2 CUN4BIPR_CTLF_Len_Req in a double-word boundary area. CUN4BIPR_CTLF_Len_Req is provided in the IDF file CUNBIIDF.

CUN4BIPR_CCSID2_CTLDEF_ALET- set by caller

Specifies the ALET for CUN4BIPR_CCSID2_CTLDEF_Ptr and is required only if CUN4BIPR_CCSID2_CTLDEF_Ptr is specified.

CUN4BIPR_CCSID2_CTLDEF_Num - set by the service

Specifies the number of entries in CUN4BIPR_CCSID2_CTLF buffer. If CUN4BIPR_CCSID2_CTLDEF_Num is equal to 1, CCSID2 is not a mixed conversion. If CUN4BIPR_CCSID2_CTLDEF_Num is > 1, CCSID2 is a mixed CCSID.

CUNBIPRM_CTLF - set by the service

CUN4BIPR_CF_CCSID - set by the service

If the input CCSID is a mixed CCSID, this specifies the sub CCSID. Otherwise, this specifies the input CCSID.

CUN4BIPR_CF_SP_STATE - set by the service

The state number of the space character in which the code point is to be used.

CUN4BIPR_CF_SP_NUM - set by the service

The number of space characters in this element for this CCSID.

CUN4BIPR_CF_SP_WIDTH - set by the service

The width of the space character code point.

CUN4BIPR_CF_SP_CODE - set by the service

The space character code point.

For UCS-2, the four bytes code point are divided in two halves. The left most two bytes are for single wide space and the right most two bytes are for double wide space.

For UTF-32, the actual length for space character is 4-bytes. However, the code point values returned are two bytes for each space character so left paddings with zeros to four bytes are needed before use. For example, for UTF-32, a value of x'0020' should be padded to 4-bytes as x'00000020' before use.

CUN4BIPR_CF_SUB_STATE - set by the service

The state number of the sub character in which the code point is to be used.

CUN4BIPR_CF_SUB_NUM - set by the service

The number of sub characters in this element for this CCSID.

CUN4BIPR_CF_SUB_WIDTH - set by the service

The width of the sub character code point.

CUN4BIPR_CF_SUB_CODE - set by the service

The substitution character code point.

For UCS-2, the four bytes are divided in two halves. The left most two bytes are used for conversions from SBCS, and the right most two bytes are for conversions from MBCS.

For UTF-32, the actual length for substitution character is 4-bytes. However, the code point values returned are two bytes for each substitution character so left paddings with zeros to four bytes are needed before use. For example, for UTF-32, a value of x'001A' and x'FFFD' should be padded to x'0000001A' and x'0000FFFF' before use.

CUN4BIPR_CF_NL_STATE - set by the service

The state number of the new line character in which the code point is to be used.

CUN4BIPR_CF_NL_WIDTH - set by the service

The width of the new line character code point.

CUN4BIPR_CF_NL_CODE - set by the service

The new line character code point.

There is only one new line character code point for each CCSID or sub CCSID, and the code point is right aligned.

CUN4BIPR_CF_LF_STATE - set by the service

The state number of the line feed character in which the code point is to be used.

CUN4BIPR_CF_LF_WIDTH - set by the service

The width of the line feed character code point.

CUN4BIPR_CF_LF_CODE - set by the service

The line feed character code point.

There is only one line feed character code point for each CCSID or sub CCSID, and the code point is right aligned.

CUN4BIPR_CF_CR_STATE - set by the service

The state number of the carriage return control character in which the code point is to be used.

CUN4BIPR_CF_CR_WIDTH - set by the service

The width of the carriage return control character code point.

CUN4BIPR_CF_CR_CODE - set by the service

The carriage return character code point.

There is only one carriage return character code point for each CCSID or sub CCSID, and the code point is right aligned.

CUN4BIPR_CF_EOF_STATE - set by the service

The state number of the end of file character in which the code point is to be used.

CUN4BIPR_CF_EOF_WIDTH - set by the service

The width of the end of file character code point.

CUN4BIPR_CF_EOF_CODE - set by the service

The end of file character code point.

There is only one end of file character code point for each CCSID or sub CCSID, and the code point is right aligned.

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.

CUN4BIPR_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.

CUN4BIPR_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.

CUN4BIPR_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_Info - set by the service**CUN4BIPR_subCCSIDs_CCSID - set by the service**

Specifies subCCSIDs.

CUN4BIPR_subCCSIDs_Size - set by the service

Specifies the size character for the subCCSID.

CUN4BIPR_subCCSIDs_Suffix - set by the service

Specifies the suffix characters 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

Chapter 10. Dynamic locale service

This information describes the z/OS Unicode Services dynamic locale service.

The z/OS Unicode Services dynamic locale service dynamically builds and loads locale data into the z/OS Unicode Services environment. These locale objects exist outside of the C/C++ Run-time storage and are available to any z/OS Unicode Services user.

The z/OS Unicode Services dynamic locale service is called using a stub routine called CUNLLOCB for AMODE (31) and CUN4LLOC for AMODE (64).

The following locale categories are supported:

- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_MONETARY
- LC_NUMERIC
- LC_TIME

Adding and removing locales in the z/OS Unicode environment

z/OS Unicode Services maintains the storage for locale objects. Once created, the locale objects remain available unless they are deleted via an operator command or at the next IPL. Additionally, the z/OS Unicode Services dynamic locale service supports the build of individual locale categories as needed, thus avoiding the overhead of building all locale categories unnecessarily.

Locale objects can be added to the z/OS Unicode Services environment by the following ways:

- Calling the z/OS Unicode Services dynamic locale service and having them created dynamically.
- Adding statements to the CUNUNIXx parmlib member specifying the locales to load during the system IPL.
- Issuing the SETUNI command to add locales to the z/OS Unicode Services environment.

Note: For optimal performance, use the CUNUNIXx parmlib member to have all locales expected to be used on the system to be built and loaded at IPL time.

Mapping of parameters in C

A header file is supplied (cunhlocb.h, SYS1.SCUNHF) that contains the mapping for the z/OS Unicode Services dynamic locale service. The following structure is used in the interface to the z/OS Unicode Services dynamic locale service.

31-bit mapping

```

/*****
/*  CUNBLPRM Structure -- Dynamic Locale Service Parameter Structure */
/*****
typedef struct tag_CUNBLPRM {           /* 31 bit */
    int      Version;                  /* Parameter Area Version */
    int      Length;                   /* Parameter Area Length */
    void     *Loc_Info_Block;          /* Locale info blk ptr */
    char     Res1[4];                  /* Reserved */
    void     *Loc_Ptr;                 /* Pointer to Locale */
    unsigned int Loc_ALET;              /* ALET for Locale */
    int      Loc_Len;                  /* Locale length */
    char     Res2[4];                  /* Reserved */
    _LC_ctype_u *LC_CTYPE_Ptr;        /* Pointer to LC_CTYPE data */
}

```

```

int          LC_CTYPE_Len;          /* Length of LC_CTYPE data          */
char         Res3[8];              /* Reserved                          */
char         _LC_ctype_u           *LC_CTYPEA_Ptr;      /* Pointer to LC_CTYPEA data        */
int          LC_CTYPEA_Len;        /* Length of LC_CTYPEA data         */
char         Res4[8];              /* Reserved                          */
char         _LC_collate_u         *LC_COLLATE_Ptr;      /* Pointer to LC_COLLATE data       */
int          LC_COLLATE_Len;       /* Length of LC_COLLATE data        */
char         Res5[8];              /* Reserved                          */
char         LC_collate_u          *LC_COLLATEA_Ptr;     /* Pointer to LC_COLLATEA data      */
int          LC_COLLATEA_Len;      /* Length of LC_COLLATEA data       */
char         Res6[8];              /* Reserved                          */
char         _LC_monetary_u        *LC_MONETARY_Ptr;     /* Pointer to LC_MONETARY data      */
int          LC_MONETARY_Len;      /* Length of LC_MONETARY data       */
char         Res7[8];              /* Reserved                          */
char         _LC_numeric_u         *LC_NUMERIC_Ptr;      /* Pointer to LC_NUMERIC data       */
int          LC_NUMERIC_Len;       /* Length of LC_NUMERIC data        */
char         Res8[8];              /* Reserved                          */
char         _LC_time_u            *LC_TIME_Ptr;         /* Pointer to LC_TIME data          */
int          LC_TIME_Len;          /* Length of LC_TIME data           */
char         Res9[8];              /* Reserved                          */
char         _LC_messages_u        *LC_MESSAGES_Ptr;     /* Pointer to LC_MESSAGES data     */
int          LC_MESSAGES_Len;      /* Length of LC_MESSAGES data       */
char         Res10[8];             /* Reserved                          */
void         *DDA_Ptr;             /* Pointer to Dynamic Data Area     */
unsigned int DDA_ALET;             /* ALET for Dynamic Data Area       */
int          DDA_Len;              /* Length of Dynamic Data Area      */
char         DSName[44];           /* Locale DS Name                   */
char         DSVol[6];            /* Locale DS Volume                  */
struct {
    int          Page_fix          : 1, /* Page fix                          */
                                     /* 0 = Not Page fix                  */
                                     /* 1 = Page fix                      */
    int          Data_fmt          : 1, /* Data format                       */
                                     /* 0 = Localedef -A format          */
                                     /* 1 = Not Localedef -A format      */
                                     /* Reserved                          */
    int          : 6;              /* FLAG Byte 1 set by caller        */
} Flags1;
char         res11[1];             /* Reserved                          */
int          Return_Code;          /* Return code                       */
int          Reason_Code;          /* Reason code                       */
}CUNBLPRM;

```

64-bit mapping

```

typedef struct tag_CUN4BLPR{       /* 64 bit                          */
    int          Version;          /* Parameter Area Version           */
    int          Length;           /* Parameter Area Length            */
    void         *Loc_Info_Block;  /* Locale info blk ptr              */
    char         Res1[8];          /* Reserved                          */
    void         *Loc_Ptr;         /* Pointer to Locale                 */
    unsigned int Loc_ALET;          /* ALET for Locale                   */
    unsigned long Loc_Len;         /* Locale length                     */
    char         Res2[8];          /* Reserved                          */
    char         _LC_ctype_u       *LC_CTYPE_Ptr;        /* Pointer to LC_CTYPE data         */
    int          LC_CTYPE_Len;     /* Length of LC_CTYPE data          */
    char         Res3[4];          /* Reserved                          */
    char         _LC_ctype_u       *LC_CTYPEA_Ptr;       /* Pointer to LC_CTYPEA data        */
    int          LC_CTYPEA_Len;    /* Length of LC_CTYPEA data         */
    char         Res4[4];          /* Reserved                          */
    char         _LC_collate_u     *LC_COLLATE_Ptr;       /* Pointer to LC_COLLATE data       */
    int          LC_COLLATE_Len;   /* Length of LC_COLLATE data        */
    char         Res5[4];          /* Reserved                          */
    char         _LC_collate_u     *LC_COLLATEA_Ptr;      /* Pointer to LC_COLLATEA data      */
    int          LC_COLLATEA_Len;  /* Length of LC_COLLATEA data       */
    char         Res6[4];          /* Reserved                          */
    char         _LC_monetary_u    *LC_MONETARY_Ptr;     /* Pointer to LC_MONETARY data      */
    int          LC_MONETARY_Len;  /* Length of LC_MONETARY data       */
    char         Res7[4];          /* Reserved                          */
    char         _LC_numeric_u     *LC_NUMERIC_Ptr;      /* Pointer to LC_NUMERIC data       */
    int          LC_NUMERIC_Len;   /* Length of LC_NUMERIC data        */
    char         Res8[4];          /* Reserved                          */
    char         _LC_time_u        *LC_TIME_Ptr;         /* Pointer to LC_TIME data          */
    int          LC_TIME_Len;      /* Length of LC_TIME data           */
    char         Res9[4];          /* Reserved                          */
    char         _LC_messages_u    *LC_MESSAGES_Ptr;     /* Pointer to LC_MESSAGES data     */
    int          LC_MESSAGES_Len;  /* Length of LC_MESSAGES data       */
    char         Res10[4];         /* Reserved                          */
    void         *DDA_Ptr;         /* Pointer to Dynamic Data Area     */
    unsigned int DDA_ALET;         /* ALET for Dynamic Data Area       */
}

```

```

int      DDA_Len;          /* Length of Dynamic Data Area */
char     DSName[44];       /* Locale DS Name */
char     DSVol[6];        /* Locale DS Volume */
struct {
    int    Page_fix      : 1, /* Page fix */
        /* 0 = Not Page fix */
        /* 1 = Page fix */
        Data_fmt        : 1, /* Data Format */
        /* 0 = Localedef -A Format */
        /* 1 = Not Localedef -A Format */
        : 6; /* Reserved */
    /* FLAG Byte 1 set by caller */
    char    res11[5];      /* Reserved */
    int     Return_Code;  /* Return code */
    int     Reason_Code;  /* Reason code */
} CUN4BLPR;

```

Mapping of parameters for AMODE (31)

An example file, CUNBLIDF, is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

Table 43. Mapping of parameters in dynamic locale service AMODE (31)

| Offset Hex | Type | Length in Bytes | Name |
|------------|-----------|-----------------|--------------------------------|
| 0 | STRUCTURE | | CUNBLPRM |
| 0 | UNSIGNED | 4 | CUNBLPRM_Version |
| 4 | UNSIGNED | 4 | CUNBLPRM_Length |
| 8 | ADDRESS | 8 | CUNBLPRM_Locale_Info_Block_Ptr |
| 10 | CHARACTER | 4 | RESERVED (UNUSED) |
| 14 | ADDRESS | 4 | CUNBLPRM_Loc_Ptr |
| 18 | UNSIGNED | 4 | CUNBLPRM_Loc_ALET |
| 1C | UNSIGNED | 4 | CUNBLPRM_Loc_Len |
| 20 | CHARACTER | 4 | RESERVED (UNUSED) |
| 24 | ADDRESS | 4 | CUNBLPRM_LC_CTYPE_Ptr |
| 28 | UNSIGNED | 4 | CUNBLPRM_LC_CTYPE_Len |
| 2C | CHARACTER | 8 | RESERVED (UNUSED) |
| 34 | ADDRESS | 4 | CUNBLPRM_LC_CTYPEA_Ptr |
| 38 | UNSIGNED | 4 | CUNBLPRM_LC_CTYPEA_Len |
| 3C | CHARACTER | 8 | RESERVED (UNUSED) |
| 44 | ADDRESS | 4 | CUNBLPRM_LC_COLLATE_Ptr |
| 48 | UNSIGNED | 4 | CUNBLPRM_LC_COLLATE_Len |
| 4C | CHARACTER | 8 | RESERVED (UNUSED) |
| 54 | ADDRESS | 4 | CUNBLPRM_LC_COLLATEA_Ptr |
| 58 | UNSIGNED | 4 | CUNBLPRM_LC_COLLATEA_Len |
| 5C | CHARACTER | 8 | RESERVED (UNUSED) |
| 64 | ADDRESS | 4 | CUNBLPRM_LC_MONETARY_Ptr |
| 68 | UNSIGNED | 4 | CUNBLPRM_LC_MONETARY_Len |

Table 43. Mapping of parameters in dynamic locale service AMODE (31) (continued)

| Offset Hex | Type | Length in Bytes | Name |
|------------|-----------|-----------------|--------------------------|
| 6C | CHARACTER | 8 | RESERVED (UNUSED) |
| 74 | ADDRESS | 4 | CUNBLPRM_LC_NUMERIC_Ptr |
| 78 | UNSIGNED | 4 | CUNBLPRM_LC_NUMERIC_Len |
| 7C | CHARACTER | 8 | RESERVED (UNUSED) |
| 84 | ADDRESS | 4 | CUNBLPRM_LC_TIME_Ptr |
| 88 | UNSIGNED | 4 | CUNBLPRM_LC_TIME_Len |
| 8C | CHARACTER | 8 | RESERVED (UNUSED) |
| 94 | ADDRESS | 4 | CUNBLPRM_LC_MESSAGES_Ptr |
| 98 | UNSIGNED | 4 | CUNBLPRM_LC_MESSAGES_Len |
| 9C | CHARACTER | 8 | RESERVED (UNUSED) |
| A4 | ADDRESS | 4 | CUNBLPRM_DDA_Buf_Ptr |
| A8 | UNSIGNED | 4 | CUNBLPRM_DDA_ALET |
| AC | UNSIGNED | 4 | CUNBLPRM_DDA_Len |
| B4 | CHARACTER | 44 | CUNBLPRM_DSName |
| F8 | CHARACTER | 6 | CUNBLPRM_DSVol |
| FE | BITSTRING | 1 | |
| | 1XXX XXXX | | CUNBLPRM_Page_Fix |
| | X1XX XXXX | | CUNBLPRM_Data_Fmt |
| | XX1X XXXX | | RESERVED (UNUSED) |
| | XXX1 XXXX | | RESERVED (UNUSED) |
| | XXXX 1XXX | | RESERVED (UNUSED) |
| | XXXX X1XX | | RESERVED (UNUSED) |
| | XXXX XX1X | | RESERVED (UNUSED) |
| | XXXX XXX1 | | RESERVED (UNUSED) |
| FF | CHARACTER | 5 | RESERVED (UNUSED) |
| 104 | UNSIGNED | 4 | CUNBLPRM_Return_code |
| 108 | UNSIGNED | 4 | CUNBLPRM_Reason_code |

Description of parameters in area CUNBLPRM

CUNBLPRM_Version - set by caller

Specifies the version of the parameter area. Use version 1.

CUNBLPRM_Length - set by caller

Specifies the length of the parameter area, in bytes. Use constant CUNBLPRM_Len.

CUNBLPRM_Loc_Info_Block_Ptr - set by caller, updated by service

Pointer to the locale information block.

CUNBLPRM_Loc_Ptr – set by caller

Pointer to where the caller wants the locale data to be copied.

CUNBLPRM_Loc_ALET - set by caller

Locale pointer ALET.

CUNBLPRM_Loc_Len - set by caller, updated by service

When initially set by the caller, it specifies the size in bytes of the storage pointed to by CUNBLPRM_Loc_Ptr, which is available to the service for locale data. When the service copies locale data into the space pointed to by CUNBLPRM_Loc_Ptr, the service updates CUNBLPRM_Loc_Len with the actual length of the locale data provided.

CUNBLPRM_LC_CTYPE_Ptr - set by service

Pointer to LC_CTYPE locale data structure if available; NULL otherwise. This area is filled in when CUNBLPRM_Data_Fmt is set to 1.

CUNBLPRM_LC_CTYPE_Len - set by service

The length of LC_CTYPE data. This area is filled in when CUNBLPRM_Data_Fmt is set to 1.

CUNBLPRM_LC_CTYPEA_Ptr - set by service

Pointer to LC_CTYPEA locale data structure if available, NULL otherwise. This area is filled in when CUNBLPRM_Data_Fmt is set to 0.

CUNBLPRM_LC_CTYPEA_Len - set by service

The length of LC_CTYPEA data. This area is filled in when CUNBLPRM_Data_Fmt is set to 0.

CUNBLPRM_LC_COLLATE_Ptr - set by service

Pointer to LC_COLLATE locale data structure if available; NULL otherwise. This area is filled in when CUNBLPRM_Data_Fmt is set to 1.

CUNBLPRM_LC_COLLATE_Len - set by service

The length of LC_COLLATE data. This area is filled in when CUNBLPRM_Data_Fmt is set to 1.

CUNBLPRM_LC_COLLATEA_Ptr - set by service

Pointer to LC_COLLATEA locale data structure if available, NULL otherwise. This area is filled in when CUNBLPRM_Data_Fmt is set to 0.

CUNBLPRM_LC_COLLATEA_Len - set by service

The length of LC_COLLATEA data. This area is filled in when CUNBLPRM_Data_Fmt is set to 0.

CUNBLPRM_LC_MONETARY_Ptr - set by service

Pointer to LC_MONETARY locale data structure if available; NULL otherwise.

CUNBLPRM_LC_MONETARY_Len - set by service

The length of LC_MONETARY data.

CUNBLPRM_LC_NUMERIC_Ptr - set by service

Pointer to LC_NUMERIC locale data structure if available; NULL otherwise.

CUNBLPRM_LC_NUMERIC_Len - set by service

The length of LC_NUMERIC data.

CUNBLPRM_LC_TIME_Ptr - set by service

Pointer to LC_TIME locale data structure if available; NULL otherwise.

CUNBLPRM_LC_TIME_Len - set by service

The length of LC_TIME data.

CUNBLPRM_LC_MESSAGES_Ptr - set by service

Pointer to LC_MESSAGES locale data structure if available; NULL otherwise.

CUNBLPRM_LC_MESSAGES_Len - set by service

The length of LC_MESSAGES data.

CUNBLPRM_DDA_Buf_Ptr - set by caller

Pointer to DDA storage.

CUNBLPRM_DDA_ALET - set by caller

The ALET for CUNBLPRM_DDA_Ptr.

CUNBLPRM_DDA_Len - set by caller

Specified the size in bytes of DDA storage.

CUNBLPRM_DSName - set by service

Specifies the optional DSName if a user wants the dynamic locale service to build a user-created locale.

CUNBLPRM_DSVol - set by caller

Specifies the optional DSVol if a user wants the dynamic locale service to build a user-created locale.

CUNBLPRM_Flags1 - set by caller**Page_fix**

Specifies whether page fixing is desired:

0

Not page fixed

1

Page fixed

Data_fmt

Specifies whether to use the localedef -A formatting for the LC_CTYPE and LC_COLLATE categories:

0

Use localedef -A formatting for the LC_CTYPE and LC_COLLATE categories.

1

Do not use localedef -A formatting for the LC_CTYPE and LC_COLLATE categories.

CUNBLPRM_Return_code - set by service

The return code returned by the z/OS Unicode Services dynamic locale service.

CUNBLPRM_Reason_code - set by service

The reason code returned by the z/OS Unicode Services dynamic locale service.

Mapping of parameters for AMODE (64)

An example file, CUN4BLID, is shipped in the SYS1.MACLIB data set and contains the length of each parameter and any boundary alignment that may be necessary.

| Table 44. Mapping of parameters in dynamic locale service AMODE (64) | | | |
|--|-----------|-----------------|--------------------------------|
| Offset Hex | Type | Length in Bytes | Name |
| 0 | STRUCTURE | | CUN4BLPR |
| 0 | UNSIGNED | 4 | CUN4BLPR_Version |
| 4 | UNSIGNED | 4 | CUN4BLPR_Length |
| 8 | ADDRESS | 8 | CUN4BLPR_Locale_Info_Block_Ptr |
| 10 | CHARACTER | 8 | RESERVED (UNUSED) |
| 18 | ADDRESS | 8 | CUN4BLPR_Loc_Ptr |
| 1C | UNSIGNED | 4 | CUN4BLPR_Loc_ALET |
| 20 | UNSIGNED | 4 | CUN4BLPR_Loc_Len |
| 24 | CHARACTER | 8 | RESERVED (UNUSED) |
| 2C | ADDRESS | 8 | CUN4BLPR_LC_CTYPE_Ptr |
| 34 | UNSIGNED | 4 | CUN4BLPR_LC_CTYPE_Len |
| 38 | CHARACTER | 4 | RESERVED (UNUSED) |
| 3C | ADDRESS | 8 | CUN4BLPR_LC_CTYPEA_Ptr |

Table 44. Mapping of parameters in dynamic locale service AMODE (64) (continued)

| Offset Hex | Type | Length in Bytes | Name |
|------------|-----------|-----------------|--------------------------|
| 44 | UNSIGNED | 4 | CUN4BLPR_LC_CTYPEA_Len |
| 48 | CHARACTER | 4 | RESERVED (UNUSED) |
| 4C | ADDRESS | 8 | CUN4BLPR_LC_COLLATE_Ptr |
| 54 | UNSIGNED | 4 | CUN4BLPR_LC_COLLATE_Len |
| 58 | CHARACTER | 4 | RESERVED (UNUSED) |
| 5C | ADDRESS | 8 | CUN4BLPR_LC_COLLATEA_Ptr |
| 64 | UNSIGNED | 4 | CUN4BLPR_LC_COLLATEA_Len |
| 68 | CHARACTER | 4 | RESERVED (UNUSED) |
| 6C | ADDRESS | 8 | CUN4BLPR_LC_MONETARY_Ptr |
| 74 | UNSIGNED | 4 | CUN4BLPR_LC_MONETARY_Len |
| 78 | CHARACTER | 4 | RESERVED (UNUSED) |
| 7C | ADDRESS | 8 | CUN4BLPR_LC_NUMERIC_Ptr |
| 84 | UNSIGNED | 4 | CUN4BLPR_LC_NUMERIC_Len |
| 88 | CHARACTER | 4 | RESERVED (UNUSED) |
| 8C | ADDRESS | 8 | CUN4BLPR_LC_TIME_Ptr |
| 94 | UNSIGNED | 4 | CUN4BLPR_LC_TIME_Len |
| 98 | CHARACTER | 4 | RESERVED (UNUSED) |
| 9C | ADDRESS | 8 | CUN4BLPR_LC_MESSAGES_Ptr |
| A4 | UNSIGNED | 4 | CUN4BLPR_LC_MESSAGES_Len |
| A8 | CHARACTER | 4 | RESERVED (UNUSED) |
| AC | ADDRESS | 8 | CUN4BLPR_DDA_Buf_Ptr |
| B4 | UNSIGNED | 4 | CUN4BLPR_DDA_ALET |
| B8 | UNSIGNED | 4 | CUN4BLPR_DDA_Len |
| BC | CHARACTER | 44 | CUN4BLPR_DSName |
| 100 | CHARACTER | 6 | CUN4BLPR_DSVol |
| 106 | BITSTRING | 1 | |
| | 1XXX XXXX | | CUN4BLPR_Page_Fix |
| | X1XX XXXX | | CUN4BLPR_Data_Fmt |
| | XX1X XXXX | | RESERVED (UNUSED) |
| | XXX1 XXXX | | RESERVED (UNUSED) |
| | XXXX 1XXX | | RESERVED (UNUSED) |
| | XXXX X1XX | | RESERVED (UNUSED) |
| | XXXX XX1X | | RESERVED (UNUSED) |
| | XXXX XXX1 | | RESERVED (UNUSED) |

Table 44. Mapping of parameters in dynamic locale service AMODE (64) (continued)

| Offset Hex | Type | Length in Bytes | Name |
|------------|-----------|-----------------|----------------------|
| 107 | CHARACTER | 5 | RESERVED (UNUSED) |
| 10C | UNSIGNED | 4 | CUN4BLPR_Return_code |
| 110 | UNSIGNED | 4 | CUN4BLPR_Reason_code |

Description of parameters in area CUN4BLPR

CUN4BLPR_Version - set by caller

Specifies the version of the parameter area. Use version 1.

CUN4BLPR_Length - set by caller

Specifies the length of the parameter area, in bytes. Use constant CUN4BLPR_Len.

CUN4BLPR_Loc_Info_Block_Ptr - set by caller, updated by service

Pointer to the locale information block.

CUN4BLPR_Loc_Ptr - set by caller

Pointer to where the caller wants the locale data to be copied.

CUN4BLPR_Loc_ALET - set by caller

Locale pointer ALET.

CUN4BLPR_Loc_Len - set by caller, updated by service

When initially set by the caller, it specifies the size in bytes of the storage pointed to by CUN4BLPR_Loc_Ptr, which is available to the service for locale data. When the service copies locale data into the space pointed to by CUN4BLPR_Loc_Ptr, the service updates CUN4BLPR_Loc_Len with the actual length of the locale data provided.

CUN4BLPR_LC_CTYPE_Ptr - set by service

Pointer to LC_CTYPE locale data structure if available; NULL otherwise. This area is filled in when CUN4BLPR_Data_Fmt is set to 1.

CUN4BLPR_LC_CTYPE_Len - set by service

The length of LC_CTYPE data. This area is filled in when CUN4BLPR_Data_Fmt is set to 1.

CUN4BLPR_LC_CTYPEA_Ptr - set by service

Pointer to LC_CTYPEA locale data structure if available; NULL otherwise. This area is filled in when CUN4BLPR_Data_Fmt is set to 0.

CUN4BLPR_LC_CTYPEA_Len - set by service

The length of LC_CTYPEA data. This area is filled in when CUN4BLPR_Data_Fmt is set to 0.

CUN4BLPR_LC_COLLATE_Ptr - set by service

Pointer to LC_COLLATE locale data structure if available; NULL otherwise. This area is filled in when CUN4BLPR_Data_Fmt is set to 1.

CUN4BLPR_LC_COLLATE_Len - set by service

The length of LC_COLLATE data. This area is filled in when CUN4BLPR_Data_Fmt is set to 1.

CUN4BLPR_LC_COLLATEA_Ptr - set by service

Pointer to LC_COLLATEA locale data structure if available; NULL otherwise. This area is filled in when CUN4BLPR_Data_Fmt is set to 0.

CUN4BLPR_LC_COLLATEA_Len - set by service

The length of LC_COLLATEA data. This area is filled in when CUN4BLPR_Data_Fmt is set to 0.

CUN4BLPR_LC_MONETARY_Ptr - set by service

Pointer to LC_MONETARY locale data structure if available; NULL otherwise.

CUN4BLPR_LC_MONETARY_Len - set by service

The length of LC_MONETARY data.

CUN4BLPR_LC_NUMERIC_Ptr - set by service

Pointer to LC_NUMERIC locale data structure if available; NULL otherwise.

CUN4BLPR_LC_NUMERIC_Len - set by service

The length of LC_NUMERIC data.

CUN4BLPR_LC_TIME_Ptr - set by service

Pointer to LC_TIME locale data structure if available; NULL otherwise.

CUN4BLPR_LC_TIME_Len - set by service

The length of LC_TIME data.

CUN4BLPR_LC_MESSAGES_Ptr - set by service

Pointer to LC_MESSAGES locale data structure if available; NULL otherwise.

CUN4BLPR_LC_MESSAGES_Len - set by service

The length of LC_MESSAGES data.

CUN4BLPR_DDA_Buf_Ptr - set by caller

Pointer to DDA storage.

CUN4BLPR_DDA_ALET - set by caller

The ALET for CUNLDPRM_DDA_Ptr.

CUN4BLPR_DDA_Len - set by caller

Specified the size in bytes of DDA storage.

CUN4BLPR_DSName - set by service

Specifies the optional DSName if a user wants the dynamic locale service to build a user-created locale.

CUN4BLPR_DSVol - set by caller

Specifies the optional DSVol if a user wants the dynamic locale service to build a user-created locale.

CUN4BLPR_Flags1 - set by caller**Page_Fix**

Specifies whether page fixing is desired:

0

Not page fixed

1

Page fixed

Data_Fmt

Specifies whether to use the localedef -A formatting for the LC_CTYPE and LC_COLLATE categories:

0

Use localedef -A formatting for the LC_CTYPE and LC_COLLATE categories.

1

Do not use localedef -A formatting for the LC_CTYPE and LC_COLLATE categories.

CUN4BLPR_Return_code - set by service

The return code returned by the z/OS Unicode Services dynamic locale service.

CUN4BLPR_Reason_code - set by service

The reason code returned by the z/OS Unicode Services dynamic locale service.

Part 3. System programmer information

Chapter 11. z/OS Unicode environment

This topic describes the z/OS Unicode environment, its key concepts, what it contains, how to work with it, and related issues.

The z/OS Unicode environment holds data required to perform conversions and support the other services provided by z/OS Unicode Services. As an example, it might hold the information required to transform character data from CCSID 00037 to CCSID 01200. This conversion data normally consists of one or more conversion tables, in this case EBCDIC to Unicode, along with their related control blocks.

Various services locate conversion data within the z/OS Unicode environment. For example, if the character conversion service is asked to translate character data from CCSID 00037 to 01200, it locates the appropriate conversion table within the Unicode environment.

Later sections in this topic describe more about how conversions are added and deleted from the Unicode environment.

Key concepts behind the z/OS Unicode environment

Life cycle

The z/OS Unicode environment is created during IPL and is available for use by all jobs. It normally stays active for the lifetime of the IPL. Even if all conversions are deleted from the environment, the z/OS Unicode environment remains.

The z/OS Unicode environment starts empty, with no conversions. The CUNUNIxX parmlib statements (which may add conversions) are applied at IPL time. Other system services may begin using z/OS Unicode Services during subsequent IPL steps. After the IPL is finished, the SETUNI and DISPLAY UNI commands can be used to modify and display the Unicode environment, and various conversion services can request dynamic loading of conversions, explained below. Generally, the z/OS Unicode environment grows until it contains all the conversions needed by the various jobs running on the system.

z/OS Unicode Services has a recovery mechanism to create a new z/OS Unicode environment if the current environment becomes damaged or unavailable. This recovery procedure is automatically invoked if damage is detected and cannot be invoked manually.

Dynamic loading

When a conversion service is requested to perform a conversion, it must first locate the correct conversion data within the z/OS Unicode environment. If the conversion data is not present, the service requests that the conversion data is "dynamically loaded" into the z/OS Unicode environment. (This is also known as "Unicode on demand.") When this happens, the service waits for the conversion to load, and then continues with the conversion. When the service is called again with the same type of conversion, it locates the conversion data within the z/OS Unicode environment and does not need to dynamically load anything.

It is recommended that most customers use dynamic loading to populate their z/OS Unicode environment.

CUNUNIxX parmlib statements

During IPL, z/OS Unicode Services processes CUNUNIxX parmlib members. (These members are specified by IEASYSxx statements or IPL parameters of the form UNI=xx.) The CUNUNIxX parmlib statements modify the z/OS Unicode environment, such as loading specific conversions. CUNUNIxX parmlib statements have the same syntax and the same effect as SETUNI command parameters. See CUNUNIxX in *z/OS MVS Initialization and Tuning Reference* for details.

Use of the CUNUNIx parmlib statements is not recommended. They are not needed because of dynamic loading.

If you already have existing CUNUNIx parmlib statements, they are still supported, and you can leave them. Note, however, that the z/OS Unicode environment can be modified (as described above) after the parmlib statements take effect.

The knowledge base

IBM-supplies a knowledge base module CUNMIKBS that describes all CCSIDs shipped with z/OS support for Unicode. See [“Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID” on page 276](#) for information on how to modify the knowledge base.

The SETUNI command

The SETUNI command modifies the z/OS Unicode environment. The functions are:

1. Add a conversion to the z/OS Unicode environment (SETUNI ADD)
2. Remove conversions from the z/OS Unicode environment (SETUNI DELETE)
3. Replace conversions in the z/OS Unicode environment (SETUNI REPLACE)
4. Compact the z/OS Unicode environment, to reclaim storage used by deleted conversions (SETUNI DELETE,INACTIVE)
5. Limit the amount of page-fixed storage available to the z/OS Unicode environment (SETUNI REALSTORAGE)
6. Load a z/OS Unicode image (SETUNI ADD,IMAGE)

For more information, see [z/OS MVS System Commands](#).

The effect of each of these SETUNI commands is:

SETUNI ADD

Adds conversions to the z/OS Unicode environment. It locates the appropriate conversion tables in data set SYS1.SCUNTB (or case conversion data in data set SYS1.SCUNLOCL). For character conversions, it also consults the knowledge base which contains information about each supported CCSID. Then it copies the required conversion data into the Unicode environment for use by the conversion services. This command has the same effect as dynamically loading a conversion into the Unicode environment. Multiple conversions may be added, one per technique letter. CCSID 01200 is handled by converting it to specific UTF-16 CCSIDs (13488, 17584, etc.).

Conversions loaded by iconv requests will not show any Syslog messages when they are loaded.

SETUNI DELETE

Removes conversions from the z/OS Unicode environment. Note however, that dynamic loading may very quickly load a new copy of the conversion. It is sometimes recommended that you delete conversions when installing service without IPL, but it is usually not necessary to delete conversions.

SETUNI REPLACE

Refreshes specific conversions by deleting and then reloading them. It is rarely necessary to replace conversions. It is sometimes recommended that you replace conversions when installing service without IPL, but it is usually not necessary to replace conversions.

SETUNI DELETE,INACTIVE

Reclaims storage from deleted conversions. It does this by re-arranging the existing conversions within the z/OS Unicode environment so that the space that had been used by deleted conversions can be used again. It is rarely necessary to delete inactive conversions.

SETUNI REALSTORAGE

Sets a maximum limit on the amount of page-fixed storage the z/OS Unicode environment is allowed to use. See the REALSTORAGE topic below for more information. As of z/OS release 1.8, conversions are not loaded into page-fixed storage by default. Use of SETUNI REALSTORAGE is not recommended.

SETUNI ADD,IMAGE

Loads all conversions contained within the specified Unicode image into the z/OS Unicode environment. As of z/OS release 1.7, Unicode images are no longer needed, but are still supported. Use of conversion images is not recommended. Use dynamic loading instead.

Some of these functions take a FORCE=YES parameter. This is to remind the operator that the function can disrupt z/OS Unicode Services callers by invalidating handles and the underlying conversion data.

Equivalent commands

The SETUNI command, SET command, CUNUNIx parmlib statements, and dynamic loading share some capabilities:

- The SET command with the UNI=xx parameter is the same as the following SETUNI command:

```
SETUNI ADD,IMAGE=CUNIMGxx,DSNAME=TEST.CUNIMG.
```

- The CUNUNIx parmlib statements are the same as the SETUNI command parameters. For example, the following commands and statements are equivalent:
 - MVS command: SETUNI ADD,FROM=37,TO=1200,TECH=RECLM
 - CUNUNIx parmlib statement: ADD,FROM=37,TO=1200,TECH=RECLM
- Dynamic loading has the same effect as the equivalent SETUNI ADD command. For example, a call to the character conversion service might have the same effect as the following command:

```
SETUNI ADD,FROM=37,TO=1200,TECH=RECLM
```

The DISPLAY UNI command

The DISPLAY UNI command shows the status of the z/OS Unicode environment. For example, it can show you which conversions are loaded or how much storage is being used.

How conversions are deleted from the z/OS Unicode environment

The SETUNI DELETE command can be used to delete specific conversions from the z/OS Unicode environment. Deleting conversions data is not recommended except when it is necessary to perform maintenance (such as activating a PTF) without an IPL.

When a conversion is deleted, the control block that anchors that conversion data is changed to indicate that the specified conversion is not present. The conversion data itself is not removed. The z/OS Unicode environment's date stamp is updated, so any handles that refer to the deleted conversion become invalid. (All handles become invalid.)

There is no synchronization between conversions that are using conversion data and the function that deletes conversion data. Any conversions that are using the conversion data at the same time it is being deleted continue running normally until they find out their conversion handle has become invalid. This situation is then handled by the "handle validation" flags in the parameter area.

Note:

1. The storage used by deleted conversions can be reclaimed by using the SETUNI DELETE,INACTIVE command.
2. The SETUNI DELETE,ALL command resets the Unicode environment to the empty state.
3. Deleted conversions can be immediately reloaded by dynamic loading.

Storage requirements

This section characterizes the amount of storage that the z/OS Unicode environment requires. This is virtual storage and most of it is typically not page-fixed. System programmers have little control over how

z/OS Unicode Services handles its own storage. z/OS Unicode Services does not use common storage, and instead allocates a common dataspace to store the z/OS Unicode environment, and manages that storage.

Topics:

- How much storage the environment is using.
- Storage required for an empty environment.
- Storage required for conversion data.
- Storage required to load a conversion image.
- Storage used by deleted conversions.

The z/OS Unicode environment stores conversion data and control structures used to locate the conversion data. Use the `DISPLAY UNI,STORAGE` command to see the amount of storage used by the current z/OS Unicode environment, and the `DISPLAY UNI,CONV` command to list the specific conversions available. Deleted conversions still take up space.

z/OS Unicode Services uses 22 pages for an empty z/OS Unicode environment. This includes two pages that describe which services are loaded and help manage the z/OS Unicode environment, plus 20 pages for a table to help locate character conversion data.

The table that helps locate character conversion data is initially 20 pages. This table is filled in as character conversion data is loaded, and all 20 pages are used in a typical customer environment. Up to 138 additional pages can be used if many conversions are loaded, but typically only a few more pages are needed.

The additional storage required for each conversion depends on what type of conversion it is:

- Character conversion. The storage required depends on the encoding scheme of the particular CCSIDs involved and if the conversion is 1-stage or 2-stage.
- Case conversion. The storage required depends on the particular conversion requested.
- Normalization.
- Collation.
- String preparation. The storage is a fixed size.

The storage required to store a particular character conversion depends on whether the conversion is 1 or 2 stages, and the number of bytes used to represent character data and other factors.

For additional information, see [“Calculating the storage needed for a conversion image” on page 265.](#)

Page-fixed (REALSTORAGE)

The Unicode dataspace is in virtual storage and competes for real storage just like any other virtual storage. Some of the conversion data is page-fixed, specifically the pages from the table that holds character conversion control structures, and any conversions that specifically were loaded into page-fixed storage.

For additional information, see [“Determining the value for the REALSTORAGE parameter” on page 267.](#)

Conversion images

z/OS Unicode Services provides a capability to create a conversion image. This image is a binary file that contains a set of predefined conversions.

It is recommended that conversion images not be used and that dynamic loading be used to populate the Unicode environment.

Prior to dynamic loading on z/OS release 1.7, a conversion image was the only way to populate the z/OS Unicode environment and their use was required. Since release 1.7, conversion images are still supported, but dynamic loading is preferred.

The DB2 conversion image

Before z/OS release 1.12, there was a special DB2 image that contained all the conversions used worldwide by DB2. This support was not needed after release 1.7 and its support for dynamic loading was removed in release 1.12.

Beginning with z/OS release 1.7, you do not need to be concerned that the DB2 pre-built image is not being loaded. This is because z/OS Unicode Services now loads conversions the first time they are requested automatically or "on demand". The system starts with an empty z/OS Unicode environment and z/OS Unicode Services loads conversions as needed. This "on demand" feature makes the DB2 pre-built image unnecessary. You can see that your conversions are being loaded by issuing the MVS command `DISPLAY UNI,CONV`.

Changes in z/OS release 1.9 make it much more likely that the DB2 pre-built image will not be loaded. Specifically, the C Run-time function `iconv()` was changed to use z/OS Unicode Services to perform its conversions. If this function is used before DB2 starts, then the DB2 pre-built image is not loaded. Unicode on demand will load conversions as needed. Many programs use the `iconv()` functions and so it is likely one of these may call `iconv` before DB2 is started.

Chapter 12. Diagnostic tools for z/OS Unicode environment errors

This section describes how the system operator can recover from errors in the z/OS Unicode environment.

This section does not cover how to recover from failing API return and reason codes. For information on those issues, see the corresponding interface.

Diagnosing Unicode environment errors

z/OS Unicode Services provides several tools to help diagnose errors in the z/OS Unicode environment, such as:

- API return codes
- Console messages
- The DISPLAY,UNI command
- The z/OS Unicode environment mapping utility (CUNMIMAP)
- Dumping the z/OS Unicode dataspace

Note: You may not need all these tools to debug a specific problem.

API return codes

Some API return and reason codes indicate problems in the z/OS Unicode environment, typically those with return codes 0xC or 0x10.

Console messages

Some messages (such as CUN4026I) indicate problems in the z/OS Unicode environment.

The DISPLAY UNI command

The DISPLAY UNI command can be used to show what conversions are loaded into the z/OS Unicode environment as well as other aspects. Use the DISPLAY UNI command to show the effects of the SETUNI ADD and SETUNI DELETE commands.

Error message are normal when attempting to load conversions that do not exist and do not necessarily indicate errors in the z/OS Unicode environment.

The z/OS Unicode environment mapping utility (CUNMIMAP)

The z/OS Unicode environment mapping utility (CUNMIMAP) helps diagnose problems with the conversion environment. The utility reads the z/OS Unicode environment (or a conversion image) and reports its content. The report content is similar to the CUN3000I messages produced by the DISPLAY UNI command, but with more details. The report shows the conversions loaded, techniques available, sub-CCSID information, where control blocks and conversion tables are stored, and more.

Note: This is a diagnostic tool. This is not a programming interface. The data and the data format given by this interface is subject to change without notice. APIs are not supplied to determine the content of the z/OS Unicode environment.

To get information about a specific character conversion, use the z/OS Unicode Services conversion information service.

There is a tool to format a character conversion table that is shipped in data set SYS1.SCUNJCL(CUNJITG1).

The CUNMIMAP utility can format either the z/OS Unicode environment or a Unicode image (created by the z/OS Unicode image generator CUNMIUTL). The jobs shown below show how to invoke the z/OS Unicode environment mapper utility (shipped in SYS1.LINKLIB(CUNMIMAP)).

To format the z/OS Unicode environment, specify PARM='ACTIVE':

```
//TESTXXX JOB (12345678), 'TEST JOB', NOTIFY=&SYSUID,
//      MSGCLASS=A, MSGLEVEL=(1,1), CLASS=A,
//      REGION=512K
//STEP1  EXEC PGM=CUNMIMAP, PARM='ACTIVE'
//SYSPRINT DD   SYSOUT=*
```

To format a Unicode image, specify PARM='FILE' and DD SYSUT1 to specify which Unicode image to format:

```
//JCLMIMAP JOB (12345678), 'TEST JOB', NOTIFY=&SYSUID,
//      MSGCLASS=A, MSGLEVEL=(1,1), CLASS=A,
//      REGION=512K
//STEP1  EXEC PGM=CUNMIMAP, PARM='FILE'
//SYSPRINT DD   SYSOUT=*
//SYSUT1  DD   DSN=MY.IMAGES(CUNIMGXX), DISP=SHR
```

The z/OS Unicode environment and Unicode images have different formats, but contain many common elements. The following is an example of part of the output of the CUNMIMAP utility:

```
Image Header Report.      01/12/2009   21:29
-----
ACTIVE.....YES
Creation time.....11/18/2009 22:07:35
Dataspace token.....80003C00000000042
Dataspace alet.....01FF000E
Dataspace size.....524287
Dataspace ttoken.....00000004000000010000000000FDA4B8
Dataspace start.....00000000
Pages used.....0
Number of UCCEs.....45
Number of top level UCCEs.31
Number of UCAEs.....0
Address of first UCCE....001D92C0
Address of first UCAE....00000000

UCCE Structure Report.      01/12/2009   21:29
-----
Address  Structure          TabPtr  TabSize Conversion
-----
000E3780 13488-00037-E          000E4000   65536 Two To One
00156000 01047-13488-L          001561C0    512 One To Two
00156B80 13488-00819-L          00157000   65536 Two To One
001A8000 13488-00850-L          001A9000   65536 Two To One
000A1580 01208-00037-E          000A1580   65536 Two To One
000A1660 01208-01200-ER          000A1660    512 UTF8 To Two
000A1740 13488-00037-E          000A2000   65536 Two To One
001254A0 13488-01047-L          00126000   65536 Two To One
```

Note: Not all the fields present in the data are formatted.

Dumping the z/OS Unicode dataspace

The content of the z/OS Unicode environment can be captured and sent to IBM for analysis. The z/OS Unicode environment is implemented by a dataspace (usually named CUNDS001) owned by ASID 1. It is also helpful to include additional data such as the LPA and common storage.

The parameters to include the z/OS Unicode dataspace in a SVC dump are as follows:

```
DSPNAME(1.CUNDS*)
```

Recovering from z/OS Unicode environment errors

z/OS Unicode Services has several mechanisms to recover from z/OS Unicode environment errors:

- Delete individual conversions
- Delete all conversions (SETUNI DELETE,ALL)
- System-initiated "reset" of the z/OS Unicode environment

Delete individual conversions

If only a few conversions have errors, use the SETUNI DELETE command to delete those conversions from the z/OS Unicode environment. The next time that conversion is required, it will be re-loaded from the data set. If the conversion in the data set has the error, that should be corrected first.

Delete all conversions

If the entire z/OS Unicode environment seems to be damaged or if many conversions are affected, use the SETUNI DELETE,ALL command to re-initialize the z/OS Unicode environment to empty. After that, conversions will be loaded as needed.

System-initiated "reset" of the z/OS Unicode environment

If z/OS Unicode Services cannot locate the z/OS Unicode environment, it attempts to reset the environment by creating a new dataspace and re-anchoring that dataspace into system control blocks. The reset z/OS Unicode environment starts out empty and conversions are loaded as needed. This procedure is rarely used and cannot be invoked manually.

Invalid conversion handles

The recovery procedure may invalidate conversion handles. Code that invokes the z/OS Unicode Services interfaces should be coded to recover from this.

Chapter 13. Manually setting up z/OS Unicode Services

This topic describes how you can set up the your system to use the z/OS Unicode Services if you want to configure the system manually.

Since release 1.7, z/OS Unicode support is configured automatically and no configuration by the user is required. If you want to configure the system manually, or are supporting an existing configuration, this topic will provide you with information.

Prerequisites

For information about z/OS hardware and software prerequisites, see [z/OS Planning for Installation](#).

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

- SYS1.LPALIB and SYS1.LINKLIB, which contain z/OS Unicode Services program modules.
- SYS1.SCUNTLB, which contains all of the z/OS Unicode Services tables shipped from IBM and must be cataloged. You cannot change the name of this data set, and it must be called SYS1.SCUNTLB.
- SYS1.SCUNLOCL, which contains all the locales of Collation services and must be cataloged. You cannot change the name of this data set, and it must be called SYS1.SCUNLOCL.
- SYS1.CSSLIB, which contains linkable stub routines.

Configuring the z/OS Unicode environment

This section describes the following configuration items:

- Updating the required parmlib members.
- Determining if you need to use the MVS Message Service (MMS).

Updating parmlib members

The parmlib members that you must update to configure the system manually are CUNUNlxx and IEASYSxx.

CUNUNlxx

CUNUNl contains information that the system needs to activate, replace, or delete z/OS 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 11, “z/OS Unicode environment,” on page 243](#) 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 Unicode services provides for Japanese translation of its messages. z/OS Unicode Services provides an English message skeleton, CUNlIENU, a Japanese message skeleton, CUNlIJPN, and a sample job CUNJIMS2 in \$CUN_MSG_DS\$. See [z/OS MVS Planning: Operations](#) for more information.

Creating the z/OS Unicode Services environment

The z/OS Unicode Services environment is created during IPL. One of the ways to populate the z/OS Unicode Environment is by loading a conversion image. This section 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.

Creating a conversion image

A conversion image is a single entity that holds all necessary information to support one callable services configuration.

Note: Starting in z/OS V1R7, the Unicode Services environment is dynamically updated when a conversion service is first requested. An conversion image is no longer needed or recommended.

A conversion image can be loaded into the system during IPL or by issuing the SET UNI or SETUNI command.

Prior to z/OS V1R7, the z/OS 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 z/OS 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, z/OS Unicode Services will load the table without requiring an IPL or disrupting the caller's request.

The new z/OS Unicode Services interfaces provided starting in V1R7 are an expanded CUNUNIx 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 z/OS Unicode Services environment by merging them into the image (duplicated conversion tables or dropped-only deltas are merged into the environment).

z/OS Unicode Services uses the following when creating the conversion image:

1. Knowledge base (supplied by IBM): describes 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. z/OS Unicode Services transforms the conversion tables into an internal format and stores them in the conversion image.
3. Input statements (either from CUNUNIx 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 [“Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID” on page 276.](#)

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 259](#).
- 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 C 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 for special use conversion

Modified for special use tables can be categorized into three main groups:

- Tables that map characters like the L tables, but for older code pages.
- Tables that map characters like the iconv() function of the C Runtime library does for converters ending with "C" (for example IBM-932C).
- Other special case mappings.

(P) Modified for printer behavior

Modified for printer behavior tables are, for EBCDIC to ASCII type mappings, where the caller wants to map both CR and LF to the same value. ('15'x and '25'x both map to '0A'x).

(O-9) User-defined conversions

User-defined conversions are supported. See [“Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID” on page 276](#).

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 265](#) and [“Collation conversion” on page 265](#).

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:

Each CONVERSION control statement defines exactly one conversion that should be generated in the conversion image. This is called 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 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:

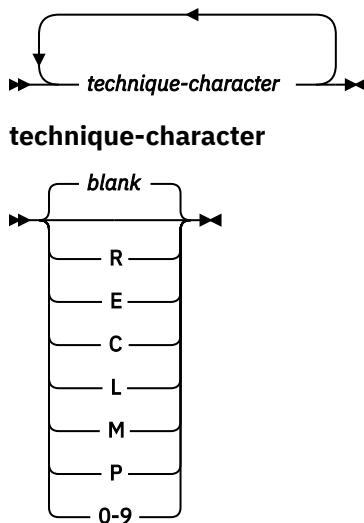
- Converting between any combination of SBCS and DBCS
- Converting between MBCS and DBCS
- Converting between UTF-8 and UCS-2

All other conversions will always be indirect conversions.

Format:

➤ CONVERSION — *from-ccsid* — , — *to-ccsid* — *technique-search-order* ; ➤

technique-search-order



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.

To support MBCS conversions, the internal techniques are used instead of the specified technique in the search order. The output of the image generator lists the table or technique that was actually selected. The internal techniques provide the equivalent support as the specified techniques and cannot be specified by customers.

Because you can specify either the default technique search order RECLM or just a blank in the CONVERSION, the field CUNBCPRM_Technique of the parameter area can contain RECLM or a blank.

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
- P: Modified for Printer 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:

- 54448 (Unicode 9.0)
- 42160 (Unicode 6.0)
- 21680 (Unicode 4.0)
- 17584 (Unicode 3.0)
- 13488 (Unicode 2.0)

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 use Roundtrip, then */
                        /* Enforced Subset */
```

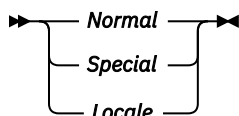
Control statement **CASE**

Purpose:

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

►► 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 SYS1.SCUNJCL contains the JCL to invoke the image generator:

```
// $JOB PREF $JOBNAME$ JOB ($ACCOUNT$), '$USER$', NOTIFY=$NOTIFY$,
//      MSGCLASS=$MC$, MSGLEVEL=$ML$, TIME=$TI$, CLASS=$CL$,
//      REGION=$REGIONØM$
// *****
// *
// * IMAGE GENERATOR
// *
// *****
// CUNMIUTL EXEC PGM=CUNMIUTL
// SYSPRINT DD SYSOUT=*
// TABIN DD DSN=$CUN_TBL_DS$, DISP=SHR
// * SYSIMG must be a FB 80 dataset *****
// SYSIMG DD DSN=$CUN_IMAGE_DS$(CUNIMGØØ), 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:

How to manually set up Unicode services

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

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 1047,850;      /* 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 CUNNCDST */
  /* compatibility decomposition table..... using CUNNKDTB */
  /* compatibility decomposition stop table. using CUNNKDST */
  /* 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 CUNOFCO */
  /* Fast Compatibility Decomposition Stop Tbl.. using CUNOFCO */
  /* 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:

```
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 TOWER     */
 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 CUNOFCO */
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 265.

Specifying the type of conversion

Use statements in the CUNUNIx 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(xxxxx) TO(yyyyy) statement. Duplicate statements are ignored. It is possible that the z/OS 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
- 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 UTF-16 version is substituted and all *technique-characters* are tested. Then the second recent UTF-16 version is substituted and so on. The supported UTF-16 CCSIDs are:

- 54448 (Unicode 9.0)
- 42160 (Unicode 6.0)
- 21680 (Unicode 4.0)
- 17584 (Unicode 3.0)
- 13488 (Unicode 2.0)

Understanding how z/OS Unicode Services loads conversion tables

When you specify one or more techniques for a particular character conversion, z/OS 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 z/OS 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 P 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. z/OS 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 the z/OS Unicode Services loads only the first table found for each sub CCSID.

CCSID 1200 is a special case since it is a virtual CCSID that represents the latest UTF-16 CCSID supported. When CCSID 1200 is specified, it is converted to the latest Unicode value supported for the conversion in question. This will result in the value of 13488, 17584, 21680, 42160, or 54448 used for the conversion.

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
- UNI600
- UNI900
- UNIX13

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
- UCA600
- UCA900
- UCAX13

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:

| <i>Table 45. Main storage needed for conversions of type SBCS and DBCS</i> | |
|--|------------------------|
| 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 46. 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 |
| MBCS→MBCS via 1200 | 320 KB |
| GB18030 MBCS→DBCS | 257 KB |
| DBCS→GB18030 MBCS | 291 KB |

If an MBCS CCSID is composed differently, break it into its sub-CCSIDs and calculate the size for each part separately, according to [Table 45 on page 265](#).

- 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 [“Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID” on page 276](#).
- For Collation tables, refer to [“Collation tables” on page 531](#).
- 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.
- For any conversion between CCSID 16684/24876 and UTF-16, an additional 2 KB is used for additional control structures.

5

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. z/OS 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, z/OS 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
                FIXED       0 PAGE
                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 CUNUNIXx 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 REALSTORAGE parameter value specifies the maximum amount of storage available for page-fixed conversion data. The z/OS Unicode environment will always have 2 pages of paged-fixed control blocks and 20 or more pages of page-fixed control data. The REALSTORAGE parameter does not control the storage used by these control blocks. It only controls and accounts for page-fixed conversion data.

The REALSTORAGE parameter does not have a minimum value. Note however that zero is a special value that does not limit the amount of page-fixed storage available. It is recommended that most installations do not specify a REALSTORAGE limit.

After invoking the image generator program to create the image, message CUN1017I, found in the listing of the //SYSPRINT DD, shows the amount of storage required to store the image in a data set. That same image when loaded into virtual memory will require additional storage. This additional storage is used by the z/OS Unicode Services internally for control structures and boundary alignment.

To calculate the value needed for the REALSTORAGE parameter, use the following formula where X is the value indicated on message CUN1017I.

```
REALSTORAGE value = (X * 1.10)
where REALSTORAGE value represents the number of pages (1 page = 4K)
```

Note:

1. Beginning with z/OS V1R7, the z/OS Unicode environment can contain additional tables that are loaded dynamically on request. These tables will take up additional storage that is not accounted for by this formula. To see the current storage used you can issue the DISPLAY UNI,STORAGE request.
2. Beginning with z/OS V1R8, the tables loaded into virtual memory, whether through dynamic load capability, contained within an image or by explicit statements in the CUNUNIXx parmlib member, are no longer page fixed by default and therefore no longer use real storage.

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 CUNUNIXx parmlib member. The z/OS 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](#).
- Use the SET UNI=xx command to specify a new CUNUNIXx 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. z/OS Unicode Services loads the tables required for conversion as they are referenced.

Note: Collation and Normalization features are not supported as part of the off-line tool CUNMIUTL for build images purposes. All collation features will be exploited as part of the z/OS Unicode Dynamic Capabilities. See [Chapter 6, "Collation," on page 111](#) and [Chapter 5, "Normalization," on page 97](#) for more information.

Chapter 14. Creating user-defined conversion tables

You can create your own user-defined conversion tables and have the z/OS Unicode Services character conversion service use them.

You might need to do this if existing conversion tables do not meet your needs, or you need to support a CCSID that is not currently supported by the z/OS Unicode Services character conversion service.

There are two different methods that can be used to customize z/OS Unicode Services:

- [“Creating a user-defined conversion table between two existing CCSIDs” on page 270](#)
- [“Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID” on page 276](#)

The method you choose to use will depend on whether you want to define a new CCSID for your user-defined conversion table.

In general, if you only want to have a different mapping occur for a subset of characters in an existing conversion table, a new CCSID is not needed and [“Creating a user-defined conversion table between two existing CCSIDs” on page 270](#) will meet your needs. One reason for creating a user-defined CCSID is because the interface you are using only allows you to pass CCSIDs and not a technique, yet you want to use a user-defined conversion. By creating a new CCSID, that value can drive new conversions.

If you think the user-defined conversion resource you need may be of use to others, contact IBM support to see if IBM can create the mapping for you.

Note: z/OS Unicode Services is used throughout z/OS. Changes to z/OS Unicode Services may impact DB/2, CICS, and so on. Because of this, you should not change or replace a system-provided mapping. Instead, create your own user-defined mapping.

Format of tables

Mapping tables exist in two formats:

- A binary format used by the system.
- A more readable text format that is easier to edit and understand, but needs to be converted to the binary format before it can be used by the system.

Table naming convention

z/OS Unicode Services ships conversion tables in binary format for its character conversion service in data set SYS1.SCUNTB. You can use these tables as a basis for your new table. These tables are based on IBM's Character Data Representation Architecture (CDRA).

In order to use the shipped conversion tables provided in data set SYS1.SCUNTB you need to understand the table naming convention. The tables shipped as members in data set SYS1.SCUNTB are named using the following naming convention: CUNtaabb.

Where:

t

is the technique character.

The technique character for tables shipped by z/OS Unicode Services in data set SYS1.SCUNTB can have the following values:

R

Roundtrip

E

Enforced subset

C

Customized

Note: This technique “C” is for customized behavior for conversion tables shipped by z/OS Unicode Services and should not be confused with user-defined conversions.

L

Language Environment-behavior

M

Modified for special use

P

Modified for printer behavior

Note:

1. Techniques 0 through 9 are reserved for customer use and are not used by z/OS Unicode Services shipped tables.
2. For more information regarding the technique character, see [Chapter 11, “z/OS Unicode environment,”](#) on page 243.

aa

is the two character suffix representing the "from" CCSID in the z/OS Unicode Services knowledge base.

bb

is the two character suffix representing the "to" CCSID in the z/OS Unicode Services knowledge base.

Example: The member CUNRCRAJ is for the roundtrip map from CCSID 500 to CCSID 256; where R = Roundtrip, CR = CCSID 500, and AJ = CCSID 256.

Note:

1. For a list of supported existing tables, see [Appendix C, “Conversion tables supplied with z/OS Unicode Services,”](#) on page 335.
2. For a list of supported CCSIDs and the table suffix to CCSID associations, see [“Encoding Scheme”](#) on page 309.
3. You cannot create a map for UTF-8 or UTF-32 conversions because these are done by converting the data to UTF-16 by a hardware instruction, then to the target.
 - If you want to modify a UTF-8 or UTF-32 mapping, you will need to update the base CCSID to or from UTF-16 map.
 - If you want to add a UTF-8 or UTF-32 mapping with the SETUNI command, you will need to reference the base UTF-16 map for the DSNNAME value.
4. For a complete list of two character table suffix/CCSID associations, see [“Step a: Select the conversions”](#) on page 255.

Creating a user-defined conversion table between two existing CCSIDs

To create a user-defined conversion table between two existing CCSIDs, the following steps are required:

- [“Step 1: Create the text format file”](#) on page 271
- [“Step 2 \(if composition characters are not on both sides\): Change the mapping”](#) on page 273
- [“Step 2 \(if composition characters are on both sides\): Change the mapping”](#) on page 273
- [“Step 3 \(if composition characters are not on both sides\): Create the binary file”](#) on page 274
- [“Step 3 \(if composition characters are on both sides\): Create the binary file”](#) on page 275
- [“Step 4: Load the new table”](#) on page 275
- [“Step 5: Update your system for the next IPL”](#) on page 276

- “Step 6: Update your program to use the new table” on page 276

User-defined technique characters are in the range 0-9 and are reserved for customer use. You need to set the technique search order when using the z/OS Unicode Services character conversion service, placing the user-defined technique character earlier in the technique search order than the technique characters for the shipped z/OS Unicode Services conversion tables.

As long as the CCSIDs involved are already defined in the z/OS Unicode Services knowledge base, you do not need to update the z/OS Unicode Services knowledge base in order to use your user-defined conversion table.

Example of building a character map based from an existing conversion table

In this example, you will take the existing CCSID 00037 to 00850 technique R mapping table and create a modified copy for your use. You will create a map that uses technique 0 and maps the '02'x value to '40'x (instead of the value of '02'x).

Step 1: Create the text format file

To create the text format file, issue this JCL:

```
//CUNMITG1 EXEC PGM=CUNMITG1,PARM='00037,00850,R'
//TABIN DD DISP=SHR,DSN=SYS1.SCUNTL
//CHAROUT DD DISP=SHR,DSN=MYDSN.TEXTMAP(MAP0AAEB)
//SYSPRINT DD SYSOUT=*
```

It will create a text map in member MAP0AAEB (specified on the CHAROUT statement) of data set MYDSN. Here is an example of the first 12 lines of the file produced:

```
01  % Character map created on 04/09/2010 at 09:54:33
02  % by CUNMITG1 Version 2.8.0
03  % Table source: CUNRAAEB
04  % Conversion mode: SBCS-SBCS
05  % Sub-character: <7F>
06  % 00037 00850
07  % -----
08  <00> <00>
09  <01> <01>
10  <02> <02>
11  <03> <03>
12  <04> <DC>
```

Additional information about using the CUNMITG1 & CUNMITG2 tools and its text file. (not required reading for the example)

The CUNMITG1 tool will create the correct output file for the inputs provided and should be used as the starting point for your customizations. Here is some more information on this tooling:

1. Although code points in the text map are sorted when created by the tool, your changed mappings can be made at any location.
2. For any duplicated code points found in the file, the last entry in the file will be the one used.
3. The acceptable techniques allowed includes 0-9 and R,E,C,L,M,P .
4. Examples of types of source files used.

SB to SB:

The previous example shows the resulting text file for a simple byte to single byte mapping as shown in the example below:

```
% -----
<00> <00>
<01> <01>
```

SB to DB:

Customization supports a single byte to double byte mapping as shown in the example below:

```
% -----  
<00> <0000>  
<01> <0001>
```

DB to SB:

Customization supports a double byte to single byte mapping as shown in the example below:

```
% -----  
<0000> <00>  
<0001> <01>
```

DB to DB:

Customization supports a double byte to double byte mappings as shown in the example below:

```
% -----  
<00A0> <D641>  
<00A1> <D642>
```

UTF-32 to DB:

Customization will also support Unicode values greater than 'FFFF'x. If this is the case, then UTF-32 notation must be used for all values in all rows for the Unicode column, as shown in the UTF-32 to double byte example below:

```
% -----  
<000000A0> <D641>  
<000000A1> <D642>  
<000100A1> <D643>
```

Composition

Customization will also support the use of composition characters in the mapping (if supported for the CCSIDs referenced). If the line contains an ampersand mark "&", this means a code point is part of a composition character and will be mapped together to create the result value.

As shown in the example below, from the CCSID 21680 to CCSID 16684 mapping, the resulting table will map '00E6'x followed by '0300'x to 'ECC3'x but will map a '00E6'x not followed by '0300'x to 'D67B'x,

```
% -----  
<000000E6> <D67B>  
<000000E6>&<00000300> <ECC3>
```

Reference information on CUNJITG1 (not required reading for the example above)

Required parameters for job CUNJITG1 are:

- PARM='from-ccsid,to-ccsid,technique' where:
 - *from-ccsid* is the source CCSID of the conversion.
 - *to-ccsid* is the target CCSID of the conversion.
 - *technique* is the technique character of the desired input conversion table.

Note:

1. Both *from-ccsid* and *to-ccsid* must be defined in the Unicode Services knowledge base prior to running job CUNJITG1.
 2. CCSID 1200 is not resolved to a particular version of Unicode Services. You have to specify a distinct Unicode CCSID instead of 1200.
 3. You must specify a distinct technique character. A technique search order is not supported here.
- //TABIN DD: Specifies the partitioned data set that holds the binary conversion tables to be used as input. This data set must be in FB 256 format.

- //CHAROUT DD: Specifies the data set that holds the created character map. This must be in FB 80 format.

Step 2 (if composition characters are not on both sides): Change the mapping

You are now ready to make the desired modifications to character map MAP0AAEB. The change desired is for an input value of '02'x to be converted to a '40'x.

To do this change line 10:

```
From: 10      <02> <02>
To:   10      <02> <40>
```

Next, add a comment to document the change.

The file would now look like this:

```
01  % Character map created on 11/01/2010 at 09:54:33
02  % Table source: CUNRAAEB
03  % Conversion mode: SBCS-SBCS
04  % Sub-character: <7F>
05  % 00037 00850
06  % -----
07  <00> <00>
08  <01> <01>
09  % Updated by Pat G. to map 02 to 40
10  <02> <40>
11  <03> <03>
12  <04> <DC>
```

Reference information on text source (not required reading for the example above)

- The % sign in the first column indicates a comment line. You can add, change or delete comment lines as desired. You can also add comments to the end of each mapping line in columns 73-80.
- Each code point that maps to a target character other than the substitution character is listed in the character map.
- The substitution character is assigned to each code point that is not explicitly provided.
- The mappings can be changed by editing the values within the < and > signs.
- You can add or delete lines from the character map.
- Each code point must be mapped on its own line and must not extend beyond a single line.
- Do not change the byte length of the character mappings. The length of the character mappings must match the length defined by the encoding scheme in the Unicode Services knowledge base. For example, because this is a single-byte to single-byte mapping, an entry of

```
08  <00> <0000>
```

is not valid since <0000> would indicate a double byte.

Step 2 (if composition characters are on both sides): Change the mapping

You are now ready to make the desired modifications to character map MAP0AAEB. The change desired is for an input value of '0201'x to be converted to a '4001'x.

To add the mapping at line 15:

```
15 <02>&<01> <40>&<01>
```

Next, add a comment to document the change.

The file would now look like this:

```
01 %
02 % Character map created on 01/04/2018 at 00:59:57
03 %                               by CUNMITG1 Version V2R4
04 %
```

```

05 % Table source: CUNRAAEB
06 % Conversion mode: SBCS-SBCS
07 % Sub-character: <7F>
08 %
09 % 00037      00850
10 % -----
11 % <00>      <00>
12 % <01>      <01>
13 % <02>      <02>
14 % Updated by somebody to map 0201 to 4001
15 % <02>&<01> <40>&<01>
16 % <03>      <03>

```

Reference information on text source (not required reading for the example above)

- '&' is used as the delimiter for composition characters.
- Each composition character length is fixed. For SBCS, it is two; for DBCS, it is four.
- At most six composition characters can be defined for each side.

Step 3 (if composition characters are not on both sides): Create the binary file

You now have a modified text file that represents the desired changes. You can use the JCL job CUNJITG2 to convert the modified text file into the binary format required to represent the mapping from CCSID 00037 to 00850 with a technique of 0.

To do this, issue the following JCL:

```

//CUNMITG2 EXEC PGM=CUNMITG2,PARM='00037,00850,0'
//CHARIN DD DISP=SHR,DSN=MYDSN.TEXTMAP(MAP0AAEB)
//TABOUT DD DISP=SHR,DSN=MYDSN.BINMAP
//SYSPRINT DD SYSOUT=*

```

This will create the binary file CUN0AAEB in MYDSN.BINMAP ready for use by Unicode Services.

Reference information on CUNJITG2 (not required reading for the example above)

Required parameters for job CUNJITG2 are:

- PARM='from-ccsid,to-ccsid,technique' where:
 - *from-ccsid* is the source CCSID of the conversion.
 - *to-ccsid* is the target CCSID of the conversion.
 - *technique* is the technique character for the output conversion table in binary format. Use a value in the range of 0-9 that is reserved for customer use.

Note:

1. Both *from-ccsid* and *to-ccsid* must be defined in the z/OS Unicode Services knowledge base prior to running job CUNJITG1.
2. CCSID 1200 is not resolved to a particular version of z/OS Unicode Services. You have to specify a distinct Unicode CCSID instead of 1200.
3. You must specify a distinct technique character. A technique search order is not supported.

Do not use alphabetic technique characters because these are reserved for z/OS Unicode Services use only. This avoids potential naming conflicts between user-defined conversion tables and those shipped by z/OS Unicode Services. It is important to avoid any possibility of naming conflicts in order to prevent the overlaying of user-defined conversion tables.

- //CHARIN DD: Specifies the data set that holds the modified character map. This must be in FB 80 format. Note that columns 73 to 80 are ignored.
- //TABOUT DD: Specifies the data set that holds the generated binary table. This must be a single data set in FB 256 format.

Step 3 (if composition characters are on both sides): Create the binary file

You now have a modified text file that represents the desired changes. You can use the JCL job CUNJITG4 to convert the modified text file into the binary format required to represent the mapping from CCSID 00037 to 00850 with a technique of 0.

To do this, issue the following JCL:

```
//CUNMITG4 EXEC PGM=CUNMITG4,PARM='00037,00850,0'
//CHARIN DD DISP=SHR,DSN=MYDSN.TEXTMAP(MAP0AAEB)
//TABOUT DD DISP=SHR,DSN=MYDSN.BINMAP
//SYSPRINT DD SYSOUT=*
```

This will create the binary file CUN0AAEB in MYDSN.BINMAP ready for use by Unicode Services.

Reference information on CUNJITG4 (not required reading for the example above)

Required parameters for job CUNJITG4 are:

- PARM='from-ccsid,to-ccsid,technique' where:
 - *from-ccsid* is the source CCSID of the conversion.
 - *to-ccsid* is the target CCSID of the conversion.
 - *technique* is the technique character for the output conversion table in binary format. Use a value in the range of 0-9 that is reserved for customer use.

Note:

1. Both *from-ccsid* and *to-ccsid* must be defined in the z/OS Unicode Services knowledge base prior to running job CUNJITG4.
2. CCSID 1200 is not resolved to a particular version of z/OS Unicode Services. You have to specify a distinct Unicode CCSID instead of 1200.
3. You must specify a distinct technique character. A technique search order is not supported.

Do not use alphabetic technique characters because these are reserved for z/OS Unicode Services use only. This avoids potential naming conflicts between user-defined conversion tables and those shipped by z/OS Unicode Services. It is important to avoid any possibility of naming conflicts in order to prevent the overlaying of user-defined conversion tables.

4. //CHARIN DD: specifies the data set that holds the modified character map. This must be in FB 80 format.

Note: Columns 73 to 80 are ignored.

5. //TABOUT DD: specifies the data set that holds the generated binary table. This must be a single data set in FB 256 format.

- //CHARIN DD: Specifies the data set that holds the modified character map. This must be in FB 80 format. Note that columns 73 to 80 are ignored.
- //TABOUT DD: Specifies the data set that holds the generated binary table. This must be a single data set in FB 256 format.

Step 4: Load the new table

Now that a table is created, it needs to be loaded so it can be used. Issue the following system command on the operator console:

```
SETUNI ADD, FROM(00037), TO(00850), TECHNIQUE(0), DSNAME(MYDSN.BINMAP)
```

Validate that this table is loaded by issuing the following DISPLAY command:

```
DISPLAY UNI,CONVERSION
```

The conversion that you just added can be seen in the list returned from the display command. In this example, you should see the highlighted entry below:

```
DISPLAY UNI,CONVERSION
CUN3000I 19.01.16 UNI DISPLAY 023
CONVERSION: 01200(13488)-00500-R      01200(13488)-53668-E
           00932-00943-RCE           00037-01200(13488)-R
           00037-00850-0
```

Step 5: Update your system for the next IPL

To ensure this table is included on future IPLs, update the CUNUNIXx parmlib member. CUNUNIXx contains information that the z/OS Unicode Services uses to define its environment. Select a CUNUNIXx parmlib member by specifying the UNI=xx keyword in IEASYSxx.

```
/*****/
/*
/* CUNUNIXX - UNICODE CONVERSION CONTROL PARAMETERS      */
/*
/*
/* updated by Pat G to load the custom table for          */
/* 00037 to 00850.                                         */
/*
*****/
ADD FROM(00037) TO(00850) TECHNIQUE(0) DSNAME(MYDSN.BINMAP)
```

Step 6: Update your program to use the new table

When your application calls the z/OS Unicode Services character conversion service, ensure that your application sets the technique search order to include the user-defined tables before other options.

In the previous example, you generated a user-defined conversion table using technique 0 so you would set the technique search order with 0 as the first technique. Therefore, you might set the technique search order to "ORECLM". This instructs Unicode Services to use the user-defined conversion table with technique character "0" if it exists prior to other tables shipped by z/OS Unicode Services.

Defining a new user-defined CCSID and then creating a user-defined conversion table using this new CCSID

This section discusses the steps necessary to create a user-defined CCSID. If you are considering creating a user-defined CCSID, contact IBM support to see if IBM already has a CCSID you can use, or if IBM can support the conversion you need.

Note: A system IPL is required to activate the newly defined CCSID.

The following steps are required to create a user-defined CCSID and then create a user-defined conversion table using this new CCSID. You will notice the steps 2 through 7 are the same as when you define a user-defined mapping in ["Creating a user-defined conversion table between two existing CCSIDs" on page 270](#).

This section will explain the steps needed for the first step only. See ["Creating a user-defined conversion table between two existing CCSIDs" on page 270](#) for steps 2 through 7.

- ["Step 1: Update the z/OS Unicode Services knowledge base" on page 277](#)
 - ["Step 1a: Modify CUNSIUKB for the new user-defined CCSID" on page 277](#)
 - ["Step 1b: Assemble and link the modified z/OS Unicode Services knowledge base module using CUNSIUKB" on page 279](#)
 - ["Step 1c: IPL the system to activate the new z/OS Unicode Services knowledge base" on page 279](#)

Once you have completed step 1, the system will support the use of the user-defined CCSID and the new table suffix in the following steps.

- ["Steps two through seven" on page 279](#)

Step 1: Update the z/OS Unicode Services knowledge base

IBM supplies a knowledge base module, CUNMIKBS, that describes all CCSIDs shipped with z/OS Unicode Services. This knowledge base tells z/OS Unicode Services how to convert each type of CCSID. CUNMIKBS is a non-executable load module stored in SYS1.LINKLIB and SYS1.LPALIB. Because CUNMIKBS is an SMP/E managed load module, it is recommended that you modify it by using an SMP/E USERMOD. User-defined CCSIDs can be added to this knowledge base using the assembler macro CUNAIKBG that is supplied in SYS1.MACLIB.

CUNSIUKB is a sample USERMOD, shipped in data set SYS1.SAMPLIB, that can be modified and used to assemble and relink module CUNMIKBS using the CUNAIKBG macro to include user-defined CCSIDs.

In the following example, a new CCSID will be created that will allow z/OS Unicode Services to pass that value on an interface and results in the user-defined conversion.

Step 1a: Modify CUNSIUKB for the new user-defined CCSID

In order to create a user-defined CCSID:

1. Choose the basic information for your new CCSID:
 - a. Choose a number from the user-defined range of 57344 to 61439 for the CCSID. The following example will use 60037 (easy to remember because it is based on codepage 37).
 - b. Choose a suffix to represent the new CCSID. The value suffix specifies a two-character alphanumeric identifier to be used in constructing the conversion table name. Unicode Services recommends picking a value in the range ZA to ZZ. The following example uses ZA.
 - c. Decide what codepage to base the new CCSID on. The following example uses 00037.
 - d. Know what values to specify for the other needed parameters. This can be discovered by looking in the CDRA documentation and copying the values for the codepage chosen in step c above. The following example uses a STYPE=1 and CCDEF=1,1,1,1,1,1 because that is what codepage 00037 uses.
2. Next, copy CUNSIUKB to another data set so you can edit the copy. In the copy of CUNSIUKB, add a comment to document the change. You should also change the lines 39 to 45 to represent the value for your new CCSID listed above. The following is an example of the default file:

```

01      ++USERMOD(UMOD001)
02      /*****
03      *
04      * Licensed Materials - Property of IBM
05      *
06      *
07      *
08      * (C) Copyright IBM Corp. 2000, 2009
09      *
10      * Status = HUN7760
11      *
12      *****/
13      *
14      * Sample usermod for building a user-defined knowledge base *
15      *
16      *****/
17      */ .
18      ++VER(Z038) FMID(HUN7760) .
19      ++JCLIN.
20      //LINK      EXEC LINKS
21      //          PARM='NCAL,MAP,LIST,LET,NOXREF,REUS',
22      //          N=,NAME=LINKLIB
23      //LINKLIB DD DSN=SYS1.LINKLIB,DISP=SHR
24      //SYSLIN DD *
25      ORDER CUNMIKBS
26      ORDER USERKBS
27      ORDER CUNMIEOF
28      MODE AMODE(31),RMODE(ANY)
29      INCLUDE LINKLIB(CUNMIKBS)
30      INCLUDE LINKLIB(USERKBS)
31      ENTRY CUNMIKBS
32      NAME CUNMIKBS(R)
33
34      ++SRC(USERKBS) DISTLIB(SCUNJCL) DISTMOD(LINKLIB) .

```

Creating user-defined conversion tables

```
35      USERKBS CSECT
36      USERKBS AMODE 31
37      USERKBS RMODE ANY
38      *
39          CUNAIKBG CCSID=57344,ES=1100,SUFFIX=ZA,CCDEF=(1,1,1,1,1,1),
40              STRINGT=1,CP=00290
41          CUNAIKBG CCSID=57345,ES=1200,SUFFIX=ZB,CCDEF=(2,2,2,2,2,2),
42              STRINGT=1,CP=00300
43          CUNAIKBG CCSID=57346,ES=1301,SUBIDS=(57344,57345),
44              STRINGT=1,CP=00290
45      END  USERKBS
```

3. Make the desired modifications in this file. Update lines 12 through 16 to add a comment to document the change.

```
12      *****
13      *
14      * Update by Pat G. to add the CCSID 60037
15      *
16      *****
```

4. Update lines 39 through 42 to reflect your new CCSIDs values and remove extra information not needed by this new CCSID.

```
37      USERKBS RMODE ANY
38      *
39          CUNAIKBG CCSID=60037,ES=1100,SUFFIX=ZA,CCDEF=(1,1,1,1,1,1),
40              STRINGT=1,CP=00037
41      END  USERKBS
```

Here is the revised file:

```
01      ++USERMOD(UMOD001)
02      /*****
03      *
04      * Licensed Materials - Property of IBM
05      *
06      *
07      *
08      * (C) Copyright IBM Corp. 2000, 2009
09      *
10      * Status = HUN7760
11      *
12      *****
13      *
14      * Updated by Pat G. to add CCSID 60037
15      *
16      *****
17      */
18      ++VER(Z038) FMID(HUN7760).
19      ++JCLIN.
20      //LINK      EXEC LINKS
21      //          PARM='NCAL,MAP,LIST,LET,NOXREF,REUS',
22      //          N=,NAME=LINKLIB
23      //LINKLIB DD DSN=SYS1.LINKLIB,DISP=SHR
24      //SYSLIN DD *
25          ORDER CUNMIKBS
26          ORDER USERKBS
27          ORDER CUNMIEOF
28          MODE AMODE(31),RMODE(ANY)
29          INCLUDE LINKLIB(CUNMIKBS)
30          INCLUDE LINKLIB(USERKBS)
31          ENTRY CUNMIKBS
32          NAME CUNMIKBS(R)
33
34      ++SRC(USERKBS) DISTLIB(SCUNJCL) DISTMOD(LINKLIB).
35      USERKBS CSECT
36      USERKBS AMODE 31
37      USERKBS RMODE ANY
38      *
39          CUNAIKBG CCSID=60037,ES=1100,SUFFIX=ZA,CCDEF=(1,1,1,1,1,1),
40              STRINGT=1,CP=00037
41      END  USERKBS
```

Step 1b: Assemble and link the modified z/OS Unicode Services knowledge base module using CUNSIUKB

You now have a modified file that represents the new CCSID you want to add to the knowledge base. See [“Reference information about the CUNAIKBG macro” on page 279](#) for the full description of all the CUNAIKBG values.

Perform an SMP/E RECEIVE and APPLY so that the source gets assembled and the load module CUNMIKBS is re-linked (contains the user-defined knowledge base CSECT USERKBS).

Reference information on CUNSIUKB

- Do not change the ORDER statements of the link step. CUNMIEOF must be the last CSECT in the load module.
- Be sure that an SMP/E ACCEPT has been performed for the z/OS Unicode Services FMID before installing the USERMOD. Otherwise, you cannot restore the original CUNMIKBS by performing an SMP/E RESTORE.

Step 1c: IPL the system to activate the new z/OS Unicode Services knowledge base

After the system IPLs, the modified knowledge base now supports the use of CCSID 60037 and the suffix of ZA for use in steps 2 through 7.

Steps two through seven

Because now you have a new user-defined CCSID, you can follow the steps in [“Creating a user-defined conversion table between two existing CCSIDs” on page 270](#). Complete the following steps in this order:

- [“Step 1: Create the text format file” on page 271](#)
- [“Step 2 \(if composition characters are not on both sides\): Change the mapping” on page 273](#)
- [“Step 3 \(if composition characters are not on both sides\): Create the binary file” on page 274](#)
- [“Step 4: Load the new table” on page 275](#)
- [“Step 5: Update your system for the next IPL” on page 276](#)
- [“Step 6: Update your program to use the new table” on page 276](#)

Reference information about the CUNAIKBG macro

The following is a description of the CUNAIKBG macro values below:

```
CUNAIKBG CCSID=60037,ES=1100,SUFFIX=ZA,CCDEF=(1,1,1,1,1,1),
        STRINGT=1,CP=00037
```

CCSID

Specifies the user-defined CCSID to be inserted into the Unicode Services 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

Specifies the encoding scheme identifier used for the CCSID. It is a four-digit identifier in hexadecimal form. For more information about encoding schemes, see [“Encoding Scheme” on page 309](#). The value es determines which of the other operands are mandatory or forbidden.

ES is required.

SUFFIX

Specifies a two-character alphanumeric identifier to be used in constructing the conversion table name. See [“Table naming convention” on page 269](#) for additional information. Modifying job CUNJIUTL suffix is required for simple CCSIDs.

SUFFIX must not be specified for mixed CCSIDs.

CCDEF

Specifies the control function definitions. These must be specified within parenthesis, separated by commas in the following order:

- sp (space)
- sub (substitute)
- nl (new line)
- lf (line feed)
- cr (carriage return)
- eof (end of line)

The values are indices into the tables described in the Character Data Representation Architecture.

CCDEF is required for simple CCSIDs. It must not be specified for mixed CCSIDs.

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 and must not be specified for simple CCSIDs.

STRINGT

The string type definition number. STRINGT is used to indicate characteristics that cannot be determined by the CCSID tag or encoding scheme alone, such as the orientation of the string or whether the characters are shaped or unshaped.

The default value is 1.

CP

Specifies the code page to be used for this entry. A code page is a specification of code points from a defined encoding scheme for each character in a set.

If you are defining a mixed CCSID, specify the code page from the single-byte component that makes up the mixed CCSID.

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 the Character Data Representation Architecture.

ACRI is required for all mixed CCSIDs, except EBCDIC MBCS. It must not be specified for simple CCSIDs and EBCDIC MBCS.

Chapter 15. Unicode batch tools

Unicode services provides tools to allow users in a batch environment to do the following:

- Access the different Unicode services provided.
- Send MIME data to mail recipients.
- Send HTML embedded multimedia data to mail recipients.

The following sections describe how to use these tools.

CUNMTUNI: Unicode Services batch client

This program uses the functions provided by the z/OS Unicode Services in batch processing.

Example

This example shows how to invoke one of the supported functions, Normalization, with CUNMTUNI.

```
//STEPEXEC PGM=CUNMTUNI
//PARMFILE DD
**request function 'normalization'
FUNC=N
*ask for
type NFCTYPE=NFC
*input is code page EBCDIC Austrian/German
CCID=1141
*insert X'0D' after reading of an input record
CRLF=1
*insert Byte Order Mark X'FEFF' to indicate UTF-16 BE
UBOM=0
/*
//DATIN DD DISP=SHR, DSN=HLQ.DS.INPUT
//DATOUT DD DISP=SHR, DSN=HLQ.DS.OUTPUT
//SYSOUT DD SYSOUT=*
```

User interface

PARMFILE: DD

Input data set containing the driving parameters to CUNMTUNI that represents the Unicode functions to be invoked as well as other parameters related with that Unicode function.

FUNC

Unicode function indicator

N

Normalization

A

Case conversion

S

Stringprep

C

Collation

V

Character conversion (including Bidirectional)

TYPE

Type of specific Unicode Service. The possible values vary with the Unicode Service, namely the value of FUNC. It is available when FUNC=N/A/S/C.

FUNC=N

NFC

Canonical composition

NFD

Canonical decomposition

NFKC

Compatibility composition

NFKD

Compatibility decomposition

FUNC=A

T

Conversion to title case.

L

Conversion to lower case

U

Conversion to upper case

FUNC=S

UTF-8

Check if input text is in UTF-8 representation.

UTF-16

Check if input text is in UTF-16 representation.

FUNC=C

<locale>

Locale for the collation

CCID

Input data CCSID indicator. This is used to convert the input data. If it is missing, the character conversion will not be performed.

TCID

Output data CCSID indicator. This parameter indicates the CCSID of the target data converted from input data in character conversion. If it is missing, it will be set to the latest Unicode version.

CRLF

Indicates the value to be inserted after reading an input record or XML data is to be processed or variable format output data set is to be used.

0

Used for EBCDIC data, and X'15' will be appended after reading a record.

1

Used for ASCII data, and X'0D' will be appended after reading a record.

2

Used for ASCII data, and X'0D0A' will be appended after reading a record.

S

XML data is to be processed. All end line characters will be removed and line is split when LRECL is exceeded or a new XML statement appears.

V

Used for EBCDIC data, and X'25' will be appended after reading a record. And variable format output data set will be used. Record is written according to the value of RECL which must be specified when CRLF=V.

RECL

Indicates the amount of bytes of one record in output data set. Valid only when CRLF=V.

UBOM

Byte Order Mask, indicates the value to be inserted before writing an output record.

0

X'FEFF' UTF-16 BE

1

X'EFBBBF' UTF-8

BKEY

Used when FUNC=V, indicates extended BIDI service is to be invoked.

Notes:

- Character conversion is performed automatically if CCID is specified, and 1200 will be the default target when you don't specify TCID.
- In Normalization and Case Conversion, either specifying CCID for input data or skipping parameter CCID to have the input encoded in 1200.
- The length of TYPE value must be less than or equal to 8.
- For Collation Service, DATOUT DD must be set as DUMMY.
- PARMFILE is usually used as an in-stream data set, otherwise, its attributes must be set as DSORG=PS, RECFM=FB, LRECL=80.

DATIN: DD

Input data set containing the data to be processed by Unicode Services.

Note: DATIN can be of any RECFM, and there is no limit on its LRECL.

DATOUT: DD

Output data set containing the processed result of input data from Unicode Service.

Note: For different CRLF values, DATOUT must be in different attributes as follows, or unexpected results can occur:

- If CRLF=S, DATOUT attributes must be DSORG=PS, RECFM=VB, LRECL=19996, BLKSIZE=20000.
- If CRLF=V, DATOUT attributes must be DSORG=PS, RECFM=V, LRECL=200.
- Otherwise, DATOUT attributes must be DSORG=PS, RECFM=U.

SYSOUT: DD

Output data set which will contain the message generated from checking the parameters in PARMFWILE.

Note:

It is usually set as SYSOUT=*, otherwise, its attributes must be DSORG=PS, RECFM=FB, BLKSIZE=13300, LRECL=80.

CUNMCSMX/CUNMRCSX: sending MIME data via SMTP

The purpose of this program is to send MIME data via SMTP. It will write user data in SMTP format into a JES-SPOOL data set. CSSMTP will handle the data set and send it as a MIME message.

The module name of this program is CUNMRCSX/CUNMCSMX.

You can both send them in a program using the CUNMCSMX or from the batch interface CUNMRCSX.

Send mail from the batch

CUNMRCSX provides the ability to send user data to mail recipients from the batch. You can send multiple MIME parts in one message to the recipients as text or binary attachments:

- EBCDIC text inline or as attachments
- EBCDIC HTML files inline or as attachments
- Binary files as attachments

The following table depicts the usable formats.

| <i>Table 47. CUNMRCSX usable formats</i> | | | |
|--|---------|-----------------|-------------------------|
| Input | Keyword | Attachment name | Output |
| EBCDIC text | TXT | - | Inline message |
| EBCDIC text | TXT | yes | 'xxxx.txt' or csv, rtf. |
| Binary | BIN | yes | pdf, doc, xls, jpg. |
| EBCDIC text | BINU | - | Inline UTF-16 |
| EBCDIC text | BINU | yes | 'xxxx.txt' or csv, rtf. |
| EBCDIC text | BIN8 | - | Inline UTF-8 |
| EBCDIC text | BIN8 | yes | 'xxxx.txt' or csv, rtf. |
| EBCDIC HTML | HTM | - | Inline HTML |
| EBCDIC HTML | HTM | yes | 'xxxx.htm' |

If you send binary data, make sure to have a format that the recipient is able to read. That can be any binary data, for example a .pdf or .jpg. Be aware that some file types may be suppressed by the recipient's mailing software.

Example

Here is an example JCL of using CUNMRCSX:

```
//STEP          EXEC  PGM=CUNMRCSX
//MIMOUT        DD    SYSOUT=(A,SMTP)
//SNAPDUMPDD    DUMMY  --if needed SYSOUT=***
//PARMDD       DD     *
HELLO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ=TEST MAIL OF CUNMRCSX
ORIG=SENDER@IBM.COM
RECP=RECPT@IBM.COM
DATA=DATA1DD  HTM
DATA=DATA2DD  TXT
//DATA1DD DD  DISP=SHR,DSN=HLQ.CNTL(DATA1DD)
//DATA2DD DD  DISP=SHR,DSN=HLQ.CNTL(DATA2DD)
```

Note: You must specify REGION=0M in your JOB or EXEC card.

User Interface

MIMOUT: DD

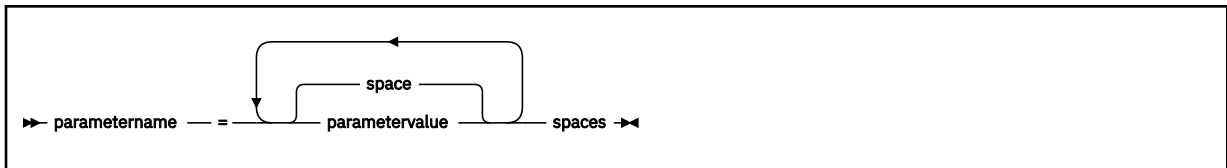
Output DD containing MIME data which will be processed by the SMTP application and sent to the recipients. For example, SYSOUT = (A, < CSSMTP JOB NAME>).

SNAPDUMP: DD

Trace DD, in case requested, please do SYSOUT=*, otherwise keep the dummy to avoid diagnostics.

PARMDD: DD

You must provide all parameters in this statement. The syntax is as follows:

**HOST**

Defines HOSTNAME, required but any value can be specified.

EHLO

Default SMTP introduction clause, preferred for CSSMTP.

HELO

Defines SMTP HELO clause as HELO. (for example, old SMTPD).

You do not need to specify these parameters for CSSMTP; EHLO is automatically added.

TRAC

Trace indicator.

Parameter can be omitted, setting default is no trace. If present, must be the second parameter. Possible values if present:

0

no trace (default)

8

maximal trace

9

extended trace (user data)

SUBJ

Subject parameter. Taken as is till up to 109 characters if available.

ORIG

Sender of a mail message (maximum length 69). Must precede DATA.

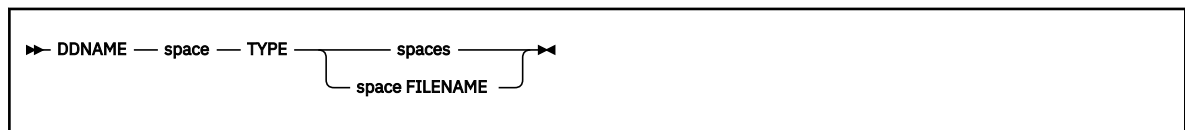
RECP

Recipient of a mail message (maximum length 69). Must precede DATA or MMSG.

For multiple recipients, specify multiple RECP statements.

DATA

Describes data to be sent. For multiple bodies, code multiple DATA statements. The syntax is as follows:

**DDNAME**

DD name you must provide in JCL as data according to example JCL.

TYPE

Corresponds to XTSONL-DATA-MIME-TYP COBOL parameter. For details please check following table.

| <i>Table 48. Data type descriptions</i> | |
|---|---|
| Value | Description |
| TXT | EBCDIC (1141) input, ASCII (923) output, CRLF added between lines. Typically output from COBOL program or text file. |
| HTM | EBCDIC (1141) input, ASCII (923) output. |
| BIN | Input is binary data, no data conversion is done, Typically encrypted data, or pre-composed hex values. |
| BINxxxxx | xxxxx stands for input CCSID, in case not present defaulted to CCSID 1141. |
| x=U | Conversion to UTF-16 is performed upon input data. |
| x=8 | Conversion to UTF-8 is performed upon input data. Example: BINU1141 to convert text from CCSID 1141 to UTF-16. |

FILENAME

If present, data will be interpreted as attachment and FILENAME will be used as name for this attachment. In case FILENAME is omitted, data is interpreted as inline text.

Return codes

The following table lists the return codes for CUNMRCSX and their descriptions:

| <i>Table 49. Return codes for CUNMRCSX</i> | |
|--|---|
| Return code | Description |
| 0 | Normal exit |
| 4 | Exit with warnings |
| 8 | Non valid parameter keyword or parameter length |
| C | Internal error |

CUNMCSMH/CUNMRCSH: sending HTML data via SMTP

This program sends HTML embedded multimedia data via SMTP. The result of the program is to write user data in SMTP format into a JES-SPOOL data set. CSSMTP will handle the data set and send it as a MIME message.

The module name of this program is 'CUNMRCSH/CUNMCSMH'.

You can either send them in a program using CUNMCSMH or from the batch interface CUNMRCSH.

Send mail from the batch

CUNMRCSH provides the ability to send HTML embedded multimedia data to mail recipients from the batch. You can send multiple MIME parts in one message to the recipients as text attachments or inline parts, as follows:

- EBCDIC HTML inline (must be the first MIME part).
- EBCDIC HTML as attachments.
- EBCDIC text as attachments.
- Binary inline files

The table below depicts the usable formats:

| Table 50. Usable formats for CUNMRCSH | | | |
|---------------------------------------|---------|-----------------|------------------------|
| Input | Keyword | Attachment name | Output |
| EBCDIC text | TXT | Yes | 'xxxx.txt' or csv, rtf |
| Binary | BIN | - | HTML embedded resource |
| EBCDIC text | BINU | Yes | 'xxxx.txt' or csv, rtf |
| EBCDIC text | BIN8 | Yes | 'xxxx.txt' or csv, rtf |
| EBCDIC HTML | HTM | - | Inline HTML |

If you send binary data, make sure to have a format that the recipient is able to read. That can be any binary data, for example a pdf or jpg. Be aware that some file types may be suppressed by the recipient's mailing software.

Example

Here is an example JCL of using CUNMRCSH:

```
//STEP          EXEC PGM=CUNMRCSH
//MIMOUT        DD SYSOUT=(A,SMTP)
//SNAPDUMP      DD DUMMY      --if needed SYSOUT=***
//PARMDD        DD *
HELO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ=TEST MAIL OF CUNMRCSH
ORIG=SENDER@IBM.COM
RECP=RECP@IBM.COM
DATA=DATA1DD HTM
DATA=DATA2DD BIN IMAGE/JPEG IMAGEID
//DATA1DD DD DISP=SHR,DSN=HLQ.CNTL(DATA1DD)
//DATA2DD DD DISP=SHR,DSN=HLQ.CNTL(DATA2DD)
```

Note:

You have to specify REGION=0M in your JOB or EXEC card.

User interface

MIMOUT: DD

Output DD containing MIME data which will be processed by the SMTP application and sent to the recipients. For example, SYSOUT = (A, <CSSMTP JOB NAME>).

SNAPDUMP: DD

Trace DD, in case requested, do SYSOUT=*, otherwise keep the dummy to avoid diagnostics.

PARMDD: DD

You must provide all parameters in this statement. The syntax is as follows:

**HOST**

Defines HOSTNAME, required but any value can be specified.

EHLO

Default SMTP introduction clause, preferred for CSSMTP.

HELO

Defines SMTP HELO clause as HELO. (for example, old SMTPD).

You do not need to specify these parameters for CSSMTP; EHLO is automatically added.

TRAC

Trace indicator.

Parameter can be omitted, setting default is no trace. If present, must be the second parameter. Possible values if present:

0

no trace (default)

8

maximal trace

9

extended trace (user data)

SUBJ

Subject parameter. Taken as is till up to 109 characters if available.

ORIG

Sender of a mail message (maximum length 69). Must precede DATA.

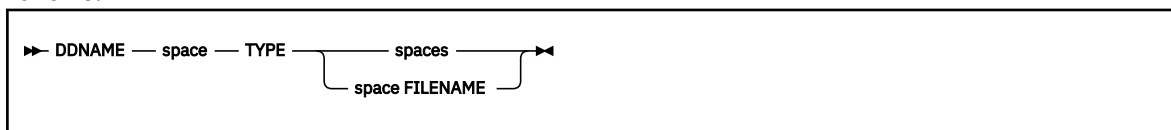
RECP

Recipient of a mail message (maximum length 69). Must precede DATA or MMSG.

For multiple recipients, specify multiple RECP statements.

DATA

Describes data to be sent. For multiple bodies, code multiple DATA statements. The syntax is as follows:

**DDNAME**

DD name you must provide in JCL as data according to example JCL.

TYPE

Corresponds to XTSONL-DATA-MIME-TYP COBOL parameter. For details please check following table.

| <i>Table 51. Data type descriptions</i> | |
|---|---|
| Value | Description |
| TXT | EBCDIC (1141) input, ASCII (923) output, CRLF added between lines. Typically output from COBOL program or text file. |
| HTM | EBCDIC (1141) input, ASCII (923) output. |
| BIN | Input is binary data, no data conversion is done, Typically encrypted data, or pre-composed hex values. |
| BINxnnnn | nnnn stands for input CCSID, in case not present defaulted to CCSID 1141. |
| x=U | Conversion to UTF-16 is performed upon input data. |
| x=8 | Conversion to UTF-8 is performed upon input data. Example: BINU1141 to convert text from CCSID 1141 to UTF-16. |

FILENAME

If present, data will be interpreted as attachment and FILENAME will be used as name for this attachment. In case FILENAME is omitted, data is interpreted as inline text.

Return codes

The following table lists the return codes for CUNMRCSH and their descriptions:

| <i>Table 52. Return codes for CUNMRCSH</i> | |
|--|---|
| Return code | Description |
| 0 | Normal exit |
| 4 | Exit with warnings |
| 8 | Non valid parameter keyword or parameter length |
| C | Internal error |

CUNMCBLI: the COBOL API utility for sending MIME data via SMTP

The purpose of this program is to allow you to write COBOL programs in order to send mail messages. The CUNMCBLI utility parses the copybook data. It then writes the output data to the JES-SPOOL data set containing the SMTP commands for one or more mail messages, which the CSSMTP application then processes.

The module name of this program is CUNMCBLI.

The JES-SPOOL data set might be freed if "FREE=CLOSE" is not specified. Therefore, if the COBOL application invokes CUNMCBLI multiple times to send multiple messages, "FREE=CLOSE" must be included, otherwise, only one message will be sent.

The copybook is defined as follows:

```

000100 01 CUNMCBLI-PARM. 00010000
000200*----- EHLO/HELO PROTOCOL -----*00020000
000300 03 CUNMCBLI-PROTOCOL PIC 9(04) 00030000
000400 VALUE 0077. 00040000
000500 88 CUNMCBLI-EHLO VALUE 0077. 00050000
000600 88 CUNMCBLI-HELO VALUE 0078. 00060000
000700*----- RET CODE SYSTEM SET -----*00070000
000800 03 CUNMCBLI-RET-CODE PIC 9(03) 00080000
000900 VALUE ZERO. 00090000
001000 88 CUNMCBLI-RET-OK VALUE ZERO. 00100000
001100 88 CUNMCBLI-RET-WRONG-PARM VALUE 047. 00110000
001200 88 CUNMCBLI-RET-ERROR-SUBCALL VALUE 067. 00120000
001300*----- MISC -----*00130000
001400 03 CUNMCBLI-MISC. 00140000
001500 05 CUNMCBLI-MISC-PRE-FILLER PIC X(56) 00150000
001600 VALUE SPACE. 00160000
001700 05 CUNMCBLI-MISC-MAX-DATA-LEN PIC 9(08) 00170000
001800 VALUE ZERO. 00180000
001900 05 CUNMCBLI-MISC-SENS-FLAG PIC X(01) 00190000
002000 VALUE SPACE. 00200000
002100 88 CUNMCBLI-MISC-SENS-FLAG-C VALUE 'C'. 00210000
002200 05 CUNMCBLI-MISC-POST-FILLER PIC X(08) 00220000
002300 VALUE SPACE. 00230000
002400*----- HOSTPARM -----*00240000
002500 03 CUNMCBLI-HOST. 00250000
002600 05 CUNMCBLI-HOST-LG PIC 9(05) 00260000
002700 VALUE 00149. 00270000
002800 05 CUNMCBLI-HOST-TYP PIC X(08) 00280000
002900 VALUE 'HOSTPARM'. 00290000
003000 05 CUNMCBLI-HOST-PORT PIC 9(05) 00300000
003100 VALUE ZERO. 00310000
003200 05 CUNMCBLI-HOST-NAME-LG PIC 9(03) 00320000
003300 VALUE 008. 00330000
003400 05 CUNMCBLI-HOST-NAME PIC X(128) 00340000
003500 VALUE 'HOSTNAME'. 00350000
003600*----- TRACPARM -----*00360000
003700 03 CUNMCBLI-TRACE. 00370000
003800 05 CUNMCBLI-TRACE-LG PIC 9(05) 00380000
003900 VALUE 00014. 00390000
004000 05 CUNMCBLI-TRACE-TYP PIC X(08) 00400000
004100 VALUE 'TRACPARM'. 00410000
004200 05 CUNMCBLI-TRACE-LVL PIC 9(01) 00420000
004300 VALUE ZERO. 00430000
004400 88 CUNMCBLI-TRACE-LVL-NONE VALUE ZERO. 00440000
004500 88 CUNMCBLI-TRACE-LVL-MAX VALUE 8. 00450000
004600 88 CUNMCBLI-TRACE-LVL-EXT VALUE 9. 00460000
004700*----- MSGHDR -----*00470000
004800* CUNMCBLI-MSGHDR-LG must be set properly. *00480000
004900* CUNMCBLI-MSGHDR-SUBJECT(120) is required. *00490000
005000* CUNMCBLI-MSGHDR-ORIG(80) is required. *00500000
005100* CUNMCBLI-MSGHDR-RECP(80) must be included at least one. *00510000
005200*-----*00520000
005300 03 CUNMCBLI-MSGHDR. 00530000
005400 05 CUNMCBLI-MSGHDR-LG PIC 9(05) 00540000
005500 VALUE 00453. 00550000
005600 05 CUNMCBLI-MSGHDR-FILLER PIC X(08) 00560000
005700 VALUE SPACE. 00570000
005800 05 CUNMCBLI-MSGHDR-SUBJECT. 00580000
005900 07 CUNMCBLI-SUBJECT PIC X(08) 00590000
006000 VALUE 'SUBJECT '. 00600000
006100 07 CUNMCBLI-SUBJECT-LG PIC 9(03) 00610000
006200 VALUE 014. 00620000
006300 07 CUNMCBLI-SUBJECT-FIELD PIC X(109) 00630000
006400 VALUE 'SUBJECT SAMPLE'. 00640000
006500 05 CUNMCBLI-MSGHDR-ORIG. 00650000
006600 07 CUNMCBLI-ORIG PIC X(08) 00660000
006700 VALUE 'FROM '. 00670000
006800 07 CUNMCBLI-ORIG-LG PIC 9(03) 00680000
006900 VALUE 015. 00690000
007000 07 CUNMCBLI-ORIG-FIELD PIC X(69) 00700000
007100 VALUE 'FROM@SOME.WHERE'. 00710000
007200 05 CUNMCBLI-MSGHDR-RECP. 00720000
007300 07 CUNMCBLI-RECP PIC X(08) 00730000
007400 VALUE 'TO '. 00740000
007500 88 CUNMCBLI-RECP-TO VALUE 'TO '. 00750000
007600 88 CUNMCBLI-RECP-CC VALUE 'CC '. 00760000
007700 88 CUNMCBLI-RECP-BCC VALUE 'BCC '. 00770000
007800 88 CUNMCBLI-RECP-REPLY-TO VALUE 'REPLY-TO'. 00780000
007900 88 CUNMCBLI-RECP-SENDER VALUE 'SENDER '. 00790000

```

```

008000      07 CUNMCBLI-RECP-LG          PIC  9(03)      00800000
008100      VALUE 014.                  00810000
008200      07 CUNMCBLI-RECP-FIELD      PIC  X(69)      00820000
008300      VALUE 'TO@OTHER.WHERE'.      00830000
010800*-----MSGDATA TEXT-----*01080000
014900      03 CUNMCBLI-MSGDATA.         01490000
015000      05 CUNMCBLI-MSGDATA-LG       PIC  9(08)      01500000
015100      VALUE ZERO.                  01510000
015200      05 CUNMCBLI-MSGDATA-TYP      PIC  X(08)      01520000
015300      VALUE 'DATA'.                 01530000
015400      05 CUNMCBLI-MSGDATA-MORE-FLAG PIC  X(01)      01540000
015500      VALUE 'E'.                    01550000
015600      88 CUNMCBLI-MORE-FLAG-M      VALUE 'M'.      01560000
015700      88 CUNMCBLI-MORE-FLAG-E      VALUE 'E'.      01570000
015800      05 CUNMCBLI-MSGDATA-ATT.     01580000
015900      07 CUNMCBLI-ATT-FLAG         PIC  X(01)      01590000
016000      VALUE 'T'.                    01600000
016100      88 CUNMCBLI-ATT-FLAG-A        VALUE 'A'.      01610000
016200      88 CUNMCBLI-ATT-FLAG-T        VALUE 'T'.      01620000
016300      07 CUNMCBLI-ATT-NAME          PIC  X(128)     01630000
016400      VALUE SPACE.                  01640000
016500      05 CUNMCBLI-MSGDATA-MIME.     01650000
016600      07 CUNMCBLI-MIME-TYP         PIC  X(03)      01660000
016700      VALUE 'TXT'.                  01670000
016800      88 CUNMCBLI-MIME-TYP-TXT      VALUE 'TXT'.     01680000
016900      88 CUNMCBLI-MIME-TYP-BIN      VALUE 'BIN'.     01690000
017000      88 CUNMCBLI-MIME-TYP-P7S      VALUE 'P7S'.     01700000
017100      88 CUNMCBLI-MIME-TYP-P7M      VALUE 'P7M'.     01710000
017200      88 CUNMCBLI-MIME-TYP-ISO      VALUE 'ISO'.     01720000
017300      88 CUNMCBLI-MIME-TYP-HTM      VALUE 'HTM'.     01730000
017400      88 CUNMCBLI-MIME-TYP-CID      VALUE 'CID'.     01740000
017500      07 CUNMCBLI-MIME-ENC          PIC  X(01)      01750000
017600      VALUE SPACE.                  01760000
017700      88 CUNMCBLI-MIME-ENC-U        VALUE 'U'.      01770000
017800      88 CUNMCBLI-MIME-ENC-8        VALUE '8'.      01780000
017900      88 CUNMCBLI-MIME-ENC-A        VALUE 'A'.      01790000
018000      07 CUNMCBLI-MIME-CCSID        PIC  X(04)      01800000
018100      VALUE SPACE.                  01810000
018200      07 CUNMCBLI-MIME-FILLER       PIC  X(56)      01820000
018300      VALUE SPACE.                  01830000
018400      05 CUNMCBLI-MSGDATA-CONTENT.  01840000
018500      07 CUNMCBLI-CONTENT-LG        PIC  9(08)      01850000
018600      VALUE 00000027.               01860000
018700      07 CUNMCBLI-CONTENT-VALUE     PIC  X(27)      01870000
018800      VALUE 'MSGDATA: TEXT INLINE SAMPLE'. 01880000

```

The description of the copybook is as follows:

| Table 53. Copybook description | | | | |
|--------------------------------|--------------------------|---------|-------|--|
| Level | Data name | Picture | Value | Description |
| 1 | CUNMCBLI-PARM | | | GROUP ITEM. The Parameter, Top level group. |
| 3 | CUNMCBLI-PROTOCOL | 9(04) | 0077 | ELEMENTARY ITEM. SMTP PROTOCOL version. 0077 EHLO (Default) 0078 HELO |
| 3 | CUNMCBLI-RET-CODE | 9(03) | ZERO | ELEMENTARY ITEM. Return code. It will be set by system after the call returns. ZERO API returns successfully. 047 Parameter wrong. 067 Internal error. |
| 3 | CUNMCBLI-MISC | | | GROUP ITEM. Defines miscellaneous options. |
| 5 | CUNMCBLI-MISC-PRE-FILLER | X(56) | SPACE | ELEMENTARY ITEM. Filler. |

Table 53. Copybook description (continued)

| Level | Data name | Picture | Value | Description |
|-------|----------------------------|---------|------------|--|
| 5 | CUNMCBLI-MISC-MAX-DATA-LEN | 9(08) | ZERO | ELEMENTARY ITEM. Maximum data Length. ZERO No limitation. Other Maximum data length. If the data length exceeded it, 047 will be returned by CUNMCBLI-RET-CODE. |
| 5 | CUNMCBLI-MISC-SENS-FLAG | X(01) | SPACE | ELEMENTARY ITEM. Send the email with sensitivity. 'C' is the only allowable value. It sets the mail as company confidential and will prohibit the mail from copying and forwarding on the client side. SPACE or any other value will be ignored. |
| 5 | CUNMCBLI-MISC-POST-FILLER | X(08) | SPACE | ELEMENTARY ITEM. Filler. |
| 3 | CUNMCBLI-HOST | | | GROUP ITEM. Defines HOST. |
| 5 | CUNMCBLI-HOST-LG | 9(05) | 00149 | ELEMENTARY ITEM. Group Length. Fixed Value. Always will be 00149 . |
| 5 | CUNMCBLI-HOST-TYP | X(08) | 'HOSTPARM' | ELEMENTARY ITEM. Group Type Indicator. Fixed Value. Always will be 'HOSTPARM' |
| 5 | CUNMCBLI-HOST-PORT | 9(05) | ZERO | ELEMENTARY ITEM. NOT used. |
| 5 | CUNMCBLI-HOST-NAME-LG | 9(03) | Number | ELEMENTARY ITEM. Length of HOST NAME. |
| 5 | CUNMCBLI-HOST-NAME | X(128) | Alphanum | ELEMENTARY ITEM. Value of HOST NAME. Maximum length is 128. |
| 3 | CUNMCBLI-TRACE | | | GROUP ITEM. Defines TRACE. |
| 5 | CUNMCBLI-TRACE-LG | 9(05) | 00014 | ELEMENTARY ITEM. Group Length. Fixed Value. Always will be 00014 . |
| 5 | CUNMCBLI-TRACE-TYP | X(08) | 'TRACPARM' | ELEMENTARY ITEM. Group Type Indicator. Fixed Value. Always will be 'TRACPARM' . |

Table 53. Copybook description (continued)

| Level | Data name | Picture | Value | Description |
|-------|-------------------------|---------|------------|---|
| 5 | CUNMCBLI-TRACE-LVL | 9(01) | ZERO | ELEMENTARY ITEM. The level of TRACE. ZERO NONE TRACE 8 MAX TRACE 9 Extended TRACE |
| 3 | CUNMCBLI-MSGHDR | | | GROUP ITEM. Defines a message. |
| 5 | CUNMCBLI-MSGHDR-LG | 9(05) | Number | ELEMENTARY ITEM. The total length of the message. |
| 5 | CUNMCBLI-MSGHDR-FILLER | X(08) | Alphanum | Filler. |
| 5 | CUNMCBLI-MSGHDR-SUBJECT | | | GROUP ITEM. Defines the subject of a message. |
| 7 | CUNMCBLI-SUBJECT | X(08) | 'SUBJECT ' | ELEMENTARY ITEM. Group Type Indicator. Fixed Value. Always will be 'SUBJECT '. |
| 7 | CUNMCBLI-SUBJECT-LG | 9(03) | Number | ELEMENTARY ITEM. The length of the subject. |
| 7 | CUNMCBLI-SUBJECT-FIELD | X(109) | Alphanum | ELEMENTARY ITEM. The value of the subject. |
| 5 | CUNMCBLI-MSGHDR-ORIG | | | GROUP ITEM. Defines the originator of a message. |
| 7 | CUNMCBLI-ORIG | X(08) | 'FROM ' | ELEMENTARY ITEM. Group Type Indicator. Fixed Value. Always will be 'FROM '. |
| 7 | CUNMCBLI-ORIG-LG | 9(03) | Number | ELEMENTARY ITEM. The length of the originator. |
| 7 | CUNMCBLI-ORIG-FIELD | X(69) | Alphanum | ELEMENTARY ITEM. The value of the originator |
| 5 | CUNMCBLI-MSGHDR-RECP | | | GROUP ITEM. Defines the recipient of a message. |
| 7 | CUNMCBLI-RECP | X(08) | 'TO ' | ELEMENTARY ITEM. Group Type Indicator. Fixed Value. It can be one of the following: <ul style="list-style-type: none"> • 'TO ' (mostly) • 'CC ' • 'BCC ' • 'REPLY-TO' • 'SENDER ' |

| Table 53. Copybook description (continued) | | | | |
|--|----------------------------|---------|------------|---|
| Level | Data name | Picture | Value | Description |
| 7 | CUNMCBLI-RECP-LG | 9(03) | Number | ELEMENTARY ITEM. The length of the recipient. |
| 7 | CUNMCBLI-RECP-FIELD | X(69) | Alphanum | ELEMENTARY ITEM. The value of the recipient. |
| 3 | CUNMCBLI-MSGDATA | | | GROUP ITEM. Defines a piece of data for a message. |
| 5 | CUNMCBLI-MSGDATA-LG | 9(08) | ZERO | ELEMENTARY ITEM. NOT used. |
| 5 | CUNMCBLI-MSGDATA-TYP | X(08) | 'DATA ' | ELEMENTARY ITEM. Group Type Indicator. Fixed Value. Always will be 'DATA ' |
| 5 | CUNMCBLI-MSGDATA-MORE-FLAG | X(01) | 'M' or 'E' | ELEMENTARY ITEM. The indicator whether this data is the last piece of the message or not. 'M' There is more data. 'E' The last piece of data. |
| 5 | CUNMCBLI-MSGDATA-ATT | | | GROUP ITEM. Defines the data as an attachment or inline text. |
| 7 | CUNMCBLI-ATT-FLAG | X(01) | 'A' or 'T' | ELEMENTARY ITEM. The indicator whether the data is attached or inline. 'A' The data will be an attachment. 'T' The data will be inline text. |
| 7 | CUNMCBLI-ATT-NAME | X(128) | Alphanum | ELEMENTARY ITEM. The value will be the attachment name if CUNMCBLI-ATT-FLAG value is 'A'. |
| 5 | CUNMCBLI-MSGDATA-MIME | | | GROUP ITEM. Defines the MIME type. |

Table 53. Copybook description (continued)

| Level | Data name | Picture | Value | Description |
|-------|-------------------|---------|-------|--|
| 7 | CUNMCBLI-MIME-TYP | X(03) | 'TXT' | <p>ELEMENTARY ITEM. MIME type indicator. The value can be one of the following:</p> <p>'TXT' EBCDIC input, ASCII output. CRLF added between lines. Both EBCDIC and ASCII CCSIDs are the defaults.</p> <p>'BIN' Input is binary, no data conversion is done if CUNMCBLI-MIME-ENC is SPACE. If the value of CUNMCBLI-MIME-ENC is 'A', 'U', '8', conversion will be done. See more information in the description of CUNMCBLI-MIME-ENC.</p> <p>'P7S' Input is binary. P7S file signed</p> <p>'P7M' Input is binary. P7M file enveloped</p> <p>'ISO' The data is already ISO-8895-15. It will be sent as it is. No conversion will be done.</p> <p>'HTM' Input is EBCDIC, Output is ASCII. Understands html and delivers inline html data, or, if filename is specified, html attachments.</p> <p>'CID' The first 'CID' points to HTML text. Subsequent 'CID' points to the embedded objects.</p> |

Table 53. Copybook description (continued)

| Level | Data name | Picture | Value | Description |
|-------|----------------------|---------|--------|--|
| 7 | CUNMCBLI-MIME-ENC | X(01) | SPACE | <p>ELEMENTARY ITEM. Determines how the output data is to be encoded.</p> <p>SPACE No Conversion</p> <p>‘U’ The output data is to be encoded in UTF-16BE. The input data encoded in EBCDIC CCSID will be converted to UTF-16BE. The EBCDIC CCSID uses the value of CUNMCBLI-MIME-CCSID if it was not zero. Otherwise, default EBCDIC CCSID will be used.</p> <p>‘8’ The output data is to be encoded in UTF-8. The input data encoded in EBCDIC CCSID will be converted to UTF-8. The EBCDIC CCSID uses the value of CUNMCBLI-MIME-CCSID if it was not zero. Otherwise, default EBCDIC CCSID will be used.</p> <p>‘A’ The output data is to be encoded in ASCII CCSID 0923 (ISO-8859-15). The input data encoded in EBCDIC CCSID will be converted to ASCII. The EBCDIC CCSID uses the value of CUNMCBLI-MIME-CCSID if it was not zero. Otherwise, default EBCDIC CCSID will be used.</p> |
| 7 | CUNMCBLI-MIME-CCSID | X(04) | Number | ELEMENTARY ITEM. The CCSID encoding for the MIME data. |
| 7 | CUNMCBLI-MIME-FILLER | X(56) | SPACE | ELEMENTARY ITEM. Filler. |

Table 53. Copybook description (continued)

| Level | Data name | Picture | Value | Description |
|-------|--------------------------|---------|----------|---|
| 5 | CUNMCBLI-MSGDATA-CONTENT | | | GROUP ITEM. Defines MSGDATA CONTENT. |
| 7 | CUNMCBLI-CONTENT-LG | 9(08) | Number | ELEMENTARY ITEM. The length of the content. |
| 7 | CUNMCBLI-CONTENT-VALUE | X(*) | Alphanum | ELEMENTARY ITEM. The value of the content. |

Example

Here is an example JCL of using CUNMCBLI:

```
//jobname JOB (accounting.information),"programmer.name",CLASS=A,MSGCLASS=A,
//          NOTIFY=userid
//APICALL EXEC PGM=MYCBLPRG
//SYSPRINT DD SYSOUT=*
//CONFIGDD DD DISP=SHR,DSN=userid.MYCBLAPI.CONFIG
//MIMOUT DD SYSOUT=(A,CSSMTP1)
```

Rules:

- If necessary, code the DCB attributes on the MIMOUT DD statement. The following example shows how to code the DCB attributes:

```
//MIMOUT DD SYSOUT=(A,CSSMTP1)
```

For the COBOL application invoking CUNMCBLI multiple times to send multiple messages, "FREE=CLOSE" must be included:

- If necessary, code the DCB attributes on the CONFIGDD DD statement. The following example shows how to code the DCB attributes:

```
//CONFIGDD DD DISP=SHR,DSN=userid.MYCBLAPI.CONFIG
```

User Interface

CSSMTP1

The external writer name if the ExtWrtName statement is specified in the CSSMTP application configuration file, or the CSSMTP application address space name if the ExtWrtName statement is not specified in the CSSMTP application configuration file.

CONFIGDD

In the DD:CONFIGDD file, the client can specify four things:

1. # DEFAULT EBCDIC CCSID WORK AS INPUT. THE DEFAULT VALUE IS 1141.
DEFAULT_EBCDIC_CCSID=#####
2. # DEFAULT ASCII CCSID WORK AS OUTPUT. THE DEFAULT VALUE IS 0923.
DEFAULT_ASCII_CCSID=#####
3. # DEFAULT MIMOUT SYSOUT CLASS, ONLY A-Z,0-9 IS ACCEPTED.
THE DEFAULT VALUE IS 'A'
MIMOUT_SYSOUT_CLASS=#
4. # DEFAULT MIMOUT CSSMTP ADDRESS SPACE NAME, 8 CHARACTERS AT MOST.
THE DEault VALUE IS 'SMTP'

```
MIMOUT_STMP_AS_NAME=#####
```

Both the SYSOUT class and CSSMTP1 defined in DD:MIMOUT statement will supersede DD:CONFIGDD option values MIMOUT_SYSOUT_CLASS and MIMOUT_STMP_AS_NAME.

If both DD:MIMOUT and DD:CONFIGDD were not used, SYSOUT default attributes will be (A,STMP). And default EBCDIC CCSID will be 1141, and default ASCII CCSID will be 0923.

CUNMCSMM/CUNMRCSM: sending MIME data via SMTP

This tool sends MIME data via SMTP. The output of this program is a MIMEOUT file, which is to be put into the JES-SPOOL. CSSMTP will handle the file and send it as an SMTP message.

The module name of this program is 'CUNMRCSM/CUNMCSMM'.

You can both send them in a program using the CUNMCSMM or from the batch interface CUNMRCSM. CUNMRCSM is an enhanced version of CUNMRCSX, which provides more capabilities.

Send mail from the batch

CUNMRCSM sends additional types of MIME data to mail recipients from the batch, compared with previous CUNMRCSX.

You can send multiple MIME parts in one message to the recipients as text attachments or inline parts:

- EBCDIC HTML inline or as attachments.
- EBCDIC Text inline or as attachments.
- Binary data embedded in HTML or as attachments
- Binary P7M data as attachments
- Binary P7S data as attachments

The table below depicts the usable formats:

| Table 54. Usable formats for CUNMRCSM | | | |
|---------------------------------------|---|-----------------|---|
| Input | Keyword | Attachment name | Output |
| EBCDIC text | TXT | - | Inline message |
| EBCDIC text | TXT | Yes | Text attachment file |
| Binary | BIN | Yes | Binary attachment file |
| EBCDIC text | BINU | - | Inline UTF-16 text file |
| EBCDIC text | BINU | Yes | UTF-16 attachment file |
| EBCDIC text | BIN8 | - | Inline UTF-8 text file |
| EBCDIC text | BIN8 | Yes | UTF-8 attachment file |
| EBCDIC text | BINA | - | Inline ASCII text file |
| EBCDIC text | BINA | Yes | ASCII attachment file |
| Binary | P7M | Yes | P7M file enveloped |
| Binary | P7S | Yes | P7S file signed |
| HTML text followed by object | CID blocks: CID CID:<multimedia type> <content location specified in CID HTML body> | - | The first 'CID' points HTML text. Subsequent 'CID' points to the embedded objects. |

If you send binary data, make sure to have a format that the recipient is able to read. That can be any binary data, for example a pdf or jpg. Be aware that some file types may be suppressed by the recipient's mailing software.

CUNMRCSM enhancements compared with CUNMRCSX

The following are CUNMRCSM enhancements compared to CUNMRCSX:

- Refers to DD: DATA=&<ddname>, optionally followed by <type> and <attachment name>.
- Same for RECP, COPY and BCPY, not for ORIG!
- CONFIGDD DD statement is added to allow users to configure the default EBCDIC and ASCII CCSIDs.
- MESSG, for short messages, max 80 bytes, you can include some system variables (&DATE, &TIME, &LPAR, &USER, &JOB, &TIMO).
- CRLF parameter to insert an additional new line feed after MESSG or DATA, unless being performed by the receiving mail system. Specify CRLF=2 before MESSG/DATA parameters in PARMDD.
- Syntax in CID statement: CID (starting from second CID) is followed by ':' and <multimedia type>, and finally by <file name>.

Sample: DATA=my.dsn CID:image/jpeg myimage.jpg.

- ATXL parameter to translate old X'7C' national substitution characters (§, Ö) to '@'. ATXL must be specified before ORIG/RECP parameters in PARMDD.
- System variables &LPAR, &DATE, &TIME, &USER, &JOB, &JNUM, &TIMO can be included in HOST, MESSG, SUBJ and attachment name.
- BINA parameter: text is translated according to UNICODE services from specified EBCDIC code page to default ASCII code page and is sent as binary in base64 format.
- Optional use of full name using the "name" <mail address> format for ORIG and SEND.
- Additional parameter RPLY for the Reply-To function.
- Additional parameter SEND for the Sender function.
- Additional parameter MAXM for the MAX MESSAGE SIZE function.
- Additional parameter VARI to set variable prefix character instead of fixed '&'.
- Additional parameter SENS to set the mail as confidential on the client side.
- CRLF parameter is extended.
- Comments in parameters are changed.

Example

Here is a JCL example using CUNMRCSM:

```
//STEP          EXEC PGM=CUNMRCSM
//MIMOUT        DD SYSOUT=(A,SMTP)
//SNAPDUMP      DD DUMMY          --if needed SYSOUT=*-
//PARMDD        DD *
HELO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ=TEST MAIL OF CUNMRCSM
ORIG=SENDER@IBM.COM
RECP=RECP@IBM.COM
DATA=&DATA1DD CID
DATA=&DATA2DD CID:IMAGE/JPEG IMAGEID
DATA=&DATA3DD TXT SAMPLE.TXT
//DATA1DD      DD DISP=SHR,DSN=HLQ.CNTL(DATA1DD)
//DATA2DD      DD DISP=SHR,DSN=HLQ.CNTL(DATA2DD)
//DATA3DD      DD DISP=SHR,DSN=HLQ.CNTL(DATA3DD)
//MIMOUT        DD SYSOUT=(A,SMTP)
```

Note: You have to specify REGION=OM in your JOB or EXEC CARD.

User interface

MIMOUT: DD

Output DD containing MIME data which will be processed by the SMTP application and sent to the recipients. For example, SYSOUT = (A, < CSSMTP JOB NAME>).

SNAPDUMP: DD

Trace DD, in case requested, do SYSOUT=*, otherwise keep the dummy to avoid diagnostics.

CONFIGDD: DD

Configuration DD, allowing users to customize the default EBCDIC and ASCII CCSID for input and output. If not provided, EBCDIC CCSID 1141 and ASCII CCSID 923 are used.

Format of configuration:

1. Lines starting with '#' are taken as Comments.
2. DEFAULT_EBCDIC_CCSID is the keyword of default EBCDIC CCSID. The value after the '=' is taken as the EBCDIC CCSID.
3. DEFAULT_ASCII_CCSID is the keyword of default EBCDIC CCSID. The value after the '=' is taken as the EBCDIC CCSID.

An example of configuration:

```
# DEFAULT EBCDIC CCSID WORK AS INPUT
DEFAULT_EBCDIC_CCSID=1047
# DEFAULT ASCII CCSID WORK AS OUTPUT
DEFAULT_ASCII_CCSID=923
```

PARMDD: DD

You must provide all parameters in this statement. The syntax is as follows:



HOST

Defines HOSTNAME, required but any value can be specified. Default: &LPAR. If &LPAR is specified, it must be starting parameter, and else it will be ignored.

EHLO

Default SMTP introduction clause, preferred for CSSMTP.

HELO

Defines SMTP HELO clause as HELO.

You do not need to specify these parameters for CSSMTP; EHLO is automatically added.

TRAC

Trace indicator.

Parameter can be omitted, setting default is no trace. If present, must be the second parameter. Possible values if present:

- 0**
no trace (default)
- 8**
maximal trace
- 9**
extended trace (user data)

SUBJ

This subject parameter maximum length is 109.

The following can be included: system variables &LPAR, &DATE, &TIME, &TIMO, &USER, &JOB, &JNUM. These variables can be used in parameters SUBJ, HOST, MESSG and attachment name.

SUBJ supports pure text input and encoded input which could be represented as follows:

```
[[PURE TEXT]*| [ENCODED FORMAT TEXT]*]*
```

The encoded input is represented as follows:

```
=?<CHARSET>?<ENCODING>?<ENCODED-TEXT>?<ENCODING>?
```

where

- CHARSET = UTF-8 or others
- ENCODING = Q(Quoted-Printable) or B(Base64)
- ENCODED-TEXT = Encoded text by <ENCODING>

For more details, please refer to RFC1342.

For example, use <UTF-8 Base64 encode> to represent the character '©'.

- The UTF-8 value for '©' is the following: 0xC2A9
- The encode UTF-8 value to Base64 is as follows: wqk=
- The compose base64 format is as follows: =?UTF-8?B?wqk=?=
- The SUBJ field should be specified as follows: SUBJ==?UTF-8?B?wqk=?= . SUBJ also supports mixed format as follows: SUBJ=sample=?UTF-8?B?wqk=?=sample

MAXM

Please specify an 8 byte number indicating the maximum message size being allowed.

If the generated message has a larger length than this value, a return code of 008 will be issued. For example: MAXM=20000000 for a limit of 20MB. If you specify a value less than 8 digits, it will be automatically padded with zeros on the left.

ORIG

Sender of a mail message (maximum length 69). Must precede DATA or MESSG, and follow SUBJ.

Sample formats:

- my.orig@domain.com
- "my name" <mail address>

RECP

Recipient of a mail message (maximum length 69). Must precede DATA or MESSG.

For multiple recipients, specify multiple RECP statements. Refers to DD statement when specifying &<ddname>, and multiple referrals allowed. For example:

```
RECP=&RECPNAME
```

pointing to

```
//RECPNAME DD *
```

```
my.recipient@domain.com
```

(multiple statements may follow)

COPY

CC recipient of a mail message (maximum length 69), must precede DATA or MESSG.

For multiple recipients, specify multiple COPY statements. Refers to DD statement when specifying &<ddname> , and multiple referrals allowed. For example:

```
COPY=&COPYNAME
```

pointing to

```
//COPYNAME DD *
```

```
my.recipient.cc@domain.com
```

(multiple statements may follow)

BCPY

BCC recipient of a mail message (maximum length 69), must precede DATA or MESG.

For multiple recipients, specify multiple BCPY statements. Refers to DD statement when specifying &<ddname>, and multiple referrals allowed. For example:

```
BCPY=&BCPYNAME
```

pointing to

```
//BCPYNAME DD *
```

```
my.recipient.bcc@domain.com
```

(multiple statements may follow)

RPLY

Reply-To recipient of a mail message (maximum length 69).

SEND

Transmits a mail message (maximum length 69).

Must be specified, when author and transmitter of a message are not identical. Name format can be either of the following:

- my.orig@domain.com
- “my name” <mail address>

VARI

Changes value for variable and address DD redirection prefix character. In some applications, the default value of & might conflict with application logic, for example OPC. You can specify the prefix as any 1-byte value (range 0x'00' to 0xFF'). Scope is from next parameter statement. It is possible to provide this parameter several times in parameters in case you need this special character to be different in subsequent parameters. Make sure to choose characters that will not cause conflicts. No check is being done for the user value of this parameter. Default value: VARI=&. For example:

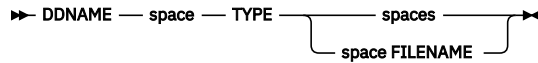
```
ORIG=&ORIGNAME  
VARI=~  
RECP=~RECPNAME
```

SENS

Send the email with sensitivity. Only SENS=C is available. It sets the mail as company confidential and will prohibit the mail from copying and forwarding in the client side.

DATA

Describes data to be sent. For multiple bodies, code multiple DATA statements. The syntax is as follows:



DDNAME

DDNAME may be replaced by a DD statement variable, for example DATA=&SENDFILE. In this case, the DD statement SENDFILE must be presented in the JCL. Temporary data sets, instream data and concatenated data must be presented this way.

TYPE

Specifies type of the data to be sent. Possible values and corresponding descriptions are shown in the following table:

| Table 55. Data type descriptions | |
|----------------------------------|--|
| Value | Description |
| TXT | EBCDIC input, ASCII output, CRLF added between lines. Typically output from COBOL programs or text file. |
| HTM | EBCDIC input, ASCII output, understands html and delivers inline html data or, if the filename is specified, html attachments. |
| BIN | Input is binary data, no data conversion is done. Typically encrypted data, or pre-composed hex values. |
| BINxnnnn | nnnn stands for input code page. If not present, defaulted to cp1141. |
| x=U, x=8, x=A | Conversion to UTF-16 is performed upon input data. Conversion to UTF-8 is performed upon input data. Conversion to ASCII is performed upon input data. For example: BINU1141 to convert text from code page 1141 to UTF-16. BINA1153 to convert text from code page 1153 to ASCII cp923. |
| P7M | Input is binary data, no data conversion is done, encrypted data in enveloped format. |
| P7S | Input is binary data, no data conversion is done, signed data in enveloped format. |
| CID | HTML data, containing location identifiers being embedded as binary files from subsequent CID statements. |

FILENAME

If present, data will be interpreted as attachment and FILENAME will be used as name for this attachment. In case FILENAME is omitted, data is interpreted as inline text.

If the CID data type is used, you must either leave the FILENAME empty (first CID value in the CID block, must be an HTML or text file specified in DDNAME), or starting from the second CID in the block followed by a multimedia type like 'image/jpeg', separated by ':', and then followed by a unique location identifier within the step, for example, a file name such as image1.jpg .

The following can be included: system variables &LPAR, &DATE, &TIME, &TIM0, &USER, &JOB, &JNUM: you can include those variables in HOST, MESS, SUBJ and attachment name.

Various formats for &DATE are available, &DATx:

- when x=0: 'YYMMDD'
- when x=1: 'MM/DD/YY'

The value for x=other is the same as the value of &DATE: 'DD.MM.YYYY'.

Various formats for &TIME are available, &TIMx:

- when x=E: 'HH:MM:SS'
- when x=0: 'HHMMSS' (accepted as part of the file name in attachments)

There are many types of multimedia files you can use, not only image/jpeg.

MESS

Describes data to be sent. Length is fixed 80 bytes, useful for short messages. Apostrophes not required.

The following can be included: system variables &LPAR, &DATE, &TIME, &TIM0 &USER, &JOB, you can include those variables in HOST, MESS, SUBJ and attachment name.

CRLF

Sets a new line sequence for the ending one; fetched record in case of text file. Scope of variable starts with next parameter. You can provide this parameter several times to change CRLF processing for different text blocks. Here are the permitted values and their descriptions:

0

Append nothing after text record.

1

Append hex'0D'; this is the default setting.

2

Append hex'0D0A'.

The following are some examples:

```
CRLF=0 RTF applications (append nothing)
DATA=&RTFDD TXT my_doc.rtf
CRLF=2 append CRLF for windows notepad
MESS=some text data line
MESS=another text data line
```

ATXL

Converts old X'7C' substitution characters in mail addresses to the @-sign. The code page must match the character of the @-sign. Italian and German characters '§' are translated to '@' by ATXL=§@. ATXL must be placed before ORIG/RECP.

Comments in PARMDD

The following table displays parameter names and where their comments may be located:

| Table 56. Comments in PARMDD | | |
|------------------------------|---------------------------|--|
| Parameter | Comment start column | Example |
| * | | * Full line comment |
| HOST | After whitespace in value | HOST=&LPAR (this is always set); host=lparname (set where job runs) |

Table 56. Comments in PARMDD (continued)

| Parameter | Comment start column | Example |
|-------------|----------------------|--|
| TRAC | 7 | TRAC=0 (no trace) |
| SUBJ | Not applicable | |
| ORIG | Not applicable* | * In case of DD redirection, comment starts after specifying the DD containing the provided address records, DD length is always 8 characters, so comment starts in column 15. |
| RECP | Not applicable* | * In case of DD redirection, comment starts after specifying the DD containing the provided address records, DD length is always 8 characters, so comment starts in column 15. |
| COPY | Not applicable* | * In case of DD redirection, comment starts after specifying the DD containing the provided address records, DD length is always 8 characters, so comment starts in column 15. |
| BCPY | Not applicable* | * In case of DD redirection, comment starts after specifying the DD containing the provided address records, DD length is always 8 characters, so comment starts in column 15. |
| RPLY | Not applicable | |
| SEND | Not applicable | |
| DATA | Not applicable | |
| HELO | 5 | HELO this is for older SMTP clients |
| EHLO | 5 | EHLO CSSMTP recommended, default |
| MESG | Not applicable | |
| ATXL | 8 | ATXL=@\$ (German at sign conversion) |
| CRLF | 7 | CRLF=2 (from now on, append both CR and LF after the record) |
| MAXM | After the whitespace | MAXM=900000 (Maximum email size 9MB) |
| VARI | | VARI=\$ (Use the dollar sign as a variable prefix) |

Return codes

The following table lists the return codes for CUNMRCSM:

| Table 57. Return codes for CUNMRCSM | |
|-------------------------------------|---|
| Return code | Description |
| 0 | Normal exit |
| 4 | Exit with warnings |
| 8 | Non valid parameter keyword or parameter length |
| C | Internal error |

Other examples

Before you begin: Before using the examples in this section, you must specify REGION=0M in your JOB or EXEC CARD.

Example 1

This example sends one short message and two data sets represented by DATASET1, DATASET2 to two recipients.

```
//STEP          EXEC PGM=CUNMRCSM
//SNAPDUMP      DD DUMMY          --if needed, SYSOUT=*--
//PARMDD       DD *
HELO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ=Testmail-onemsg-2dsns-2recp
ORIG=SENDER@IBM.COM
RECP=RECPTER1@IBM.COM
RECP=RECPTER2@IBM.COM
MSG=Hello world, here is &USER on &LPAR at &DATE &TIME
DATA=&DATA1DD BINU1141 attmnt_1.txt
DATA=&DATA2DD BIN8 attmnt_2.txt
//DATA1DD      DD DISP=SHR,DSN=HLQ.CNTL(DATA1DD)
//DATA2DD      DD DISP=SHR,DSN=HLQ.CNTL(DATA2DD)
//MIMOUT       DD SYSOUT=(A,SMTP)
```

Example 2

This example sends one short message finished with an additional line feed. CRLF must be placed before MSG, and the replied address is specified in RPLY.

```
//STEP          EXEC PGM=CUNMRCSM
//SNAPDUMP      DD DUMMY          --if needed, SYSOUT=*--
//PARMDD       DD *
HELO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ= Testmail-onemsg-CRLF
ORIG=SENDER@IBM.COM
RECP=RECPTER@IBM.COM
CRLF=2
MSG=Hello world, here is &USER on &LPAR at &DATE &TIME
//MIMOUT       DD SYSOUT=(A,SMTP)
```

Example 3

This example sends one short message finished with an additional line feed. Old German/Italian substitution character '\$' will be translated to '@'. ATXL must be placed before the addresses.

```
//STEP          EXEC PGM=CUNMRCSM
//SNAPDUMP      DD DUMMY          --if needed, SYSOUT=*--
//PARMDD       DD *
HELO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ=Testmail-onemsg-ATXL
ATXL=$@
ORIG=SENDER$IBM.COM
RECP=RECPTER$IBM.COM
CRLF=2
MSG=Hello world, here is &USER on &LPAR at &DATE &TIME
//MIMOUT       DD SYSOUT=(A,SMTP)
```

Example 4

This example sends data from a DD statement DDX referred to in the DATA parameter. The data will be sent as text attachment A&DATE.txt.

Optionally, both TXT and attachment names could be omitted, if normal text is requested only.

Recipients are listed in the DD statement MYRECIPS.

```
//STEP          EXEC PGM=CUNMRCSM
//SNAPDUMP      DD DUMMY          --if needed, SYSOUT=*--
//PARMDD       DD *
HELO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ=Testmail-DD
ORIG=SENDER@IBM.COM
RECP=&MYRECIPS
DATA=&DDX      TXT      A&DATE.txt
//DDX          DD DISP=SHR,DSN=HLQ.CNTL(DDX)
//MYRECIPS      DD DISP=SHR,DSN=HLQ.CNTL(RECIPS)
//MIMOUT       DD SYSOUT=(A,SMTP)
```

Note:

ORIG does not point to a DD statement.

Example 5

This example shows how to send multimedia data in an inline mail body using CUNMRCSM.

In order to do this, you have to compose a block consisting of an html input followed by binary files to be imbedded.

```
//STEP          EXEC PGM=CUNMRCSM
//SNAPDUMP      DD DUMMY          --if needed, SYSOUT=*--
//PARMDD       DD *
HELO
HOST=SMTP.IBM.COM
TRAC=8
SUBJ=Testmail-HTML-multimedia
ORIG=SENDER@IBM.COM
RECP=RECPTER@IBM.COM
DATA=MY.HTML.DSN1 CID          the introducing HTML
DATA=MY.BIN.DSN2 CID:image/jpeg IMAGEID  embedded image
//MIMOUT       DD SYSOUT=(A,SMTP)
```

Note:

The data must be in a format which HTML is able to handle. In this case, it must be binary data. For the introduction HTML, it is EBCDIC text. For all other multimedia data, binary format is usually required. For CSV data, convert your data to ASCII before using it.

Appendix A. Description of CCSIDs

Unicode CCSIDs

z/OS Unicode Services supports several different CCSID values for the Unicode Standard, and they are listed here for easy reference. (It is suggested to use 1200 for general Unicode because it will default to the most current version supported.)

| CCSID | Description | Suffix |
|--------------|---|-------------------------|
| 01200 | Unicode - most recent version supported, UTF-16 encoding. | (Suffix not applicable) |
| 01202 | Unicode - most recent version supported, UTF-16. Data is little endian order. | (Suffix not applicable) |
| 01208 | Unicode - most recent version supported, UTF-8 encoding. | (Suffix not applicable) |
| 01210 | Unicode - most recent version supported, UTF-EBCDIC encoding. | UH |
| 01232 | Unicode - most recent version supported, UTF-32 encoding. | (Suffix not applicable) |
| 13488 | Unicode - version 2.0. | PG |
| 17584 | Unicode - version 3.0. | PH |
| 21680 | Unicode - version 4.0. | TH |
| 42160 | Unicode - version 6.0. | UR |
| 54448 | Unicode - version 9.0. | VA |

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.

Encoding Scheme

A basic feature of a CCSID is its encoding scheme, which is uniquely identified by the hexadecimal encoding scheme identifier (ESID). This is a summary of encoding schemes.

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.

In this topic, the following descriptions are used for the encoding schemes:

| Table 58. Encoding schemes | |
|----------------------------|--|
| ES ID Hex | ES description |
| 1100 | EBCDIC, SBCS |
| 1200 | EBCDIC, DBCS |
| 1301 | EBCDIC, Mixed single-byte and double-byte, using SO/SI code extension method |
| 6100 | EBCDIC Presentation, SBCS |
| 1808 | UTF-EBCDIC encoding |
| 7200 | UTF-16, Unicode standard UTF-16. Data is big endian order |

| <i>Table 58. Encoding schemes (continued)</i> | |
|---|---|
| ES ID Hex | ES description |
| 720B | UTF-16 LE, Unicode standard UTF-16. Data is little endian order |
| 7500 | UTF-32, Unicode standard UTF-32. Data is big endian order |
| 7807 | UTF-8, Unicode standard UTF-8 |
| 8200 | Unicode display |
| 2100 | IBM-PC Data, SBCS |
| 2200 | IBM-PC Data, DBCS |
| 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 |
| 3100 | IBM-PC Display, SBCS |
| 3200 | IBM-PC Display, DBCS |
| 3300 | IBM-PC Display, Mixed single-byte and double-byte, with implicit code extension |
| 4403 | IBM EUC |
| 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 |
| 5200 | ISO 7 (ASCII code), DBCS |
| 5700 | ISO 7 Triple-Byte Code Set |
| 5404 | ISO 2022 TCP/IP using ESC sequences |
| 5409 | ISO 2022 TCP/IP using SO/SI |
| 540A | ISO 2022 TCP/IP using SO, SI, SS2, SS3 |
| 8100 | 8 bit, SBCS, used with a 7-bit code page |
| 9200 | 8 bit, DBCS, used with a 7-bit code page |
| 2900 | PC Data, fixed 4-byte |
| 2A00 | PC Data, mixed single-byte, double-byte, four-byte |

The following table describes the CCSIDs supported by z/OS Unicode.

| <i>Table 59. CCSIDs supported by z/OS Unicode</i> | | | |
|---|------------------------|--|---------------|
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|----------------------------------|--------|
| 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 |
| 00367 | ASCII, SBCS | USA, ANSI X3.4 ASCII STANDARD | 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 |
| 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 |
| 00848 | ASCII, SBCS | UKRAINE | D7 |
| 00849 | ASCII, SBCS | BELARUS | D9 |
| 00850 | ASCII, SBCS | LATIN-1 | EB |
| 00851 | ASCII, SBCS | GREEK | EG |

| <i>Table 59. CCSIDs supported by z/OS Unicode (continued)</i> | | | |
|---|------------------------|--|---------------|
| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
| 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 |
| 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 |
| 00876 | ASCII, SBCS | OCR (OPTICAL CHARACTER RECOGNITION) | UF |
| 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 |
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|--------------------------------------|--------|
| 00915 | ASCII, SBCS | CYRILLIC, 8-BIT, ISO 8859-5 | H4 |
| 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 |
| 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 |
| 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 |

| <i>Table 59. CCSIDs supported by z/OS Unicode (continued)</i> | | | |
|---|------------------------|---|---------------|
| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
| 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 |
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|------------------------------|--------|
| 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 |
| 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 / FRENCH | 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 |
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|---|--------|
| 01158 | EBCDIC, SBCS | UKRAINE | OK |
| 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 |
| 01175 | EBCDIC, SBCS | EBCDIC Turkey with Euro and Turkish Lira | U0 |
| 01200 | UTF-16 | UTF-16 BE with IBM PUA | PF |
| 01202 | UTF-16 LE | UTF-16 LE WITH IBM PUA | T7 |
| 01208 | UTF-8 | UTF-8 WITH IBM PUA | PK |
| 01210 | UTF-EBCDIC | UTF-EBCDIC with IBM PUA | UH |
| 01232 | UTF-32 | UTF-32 BE WITH IBM PUA | J1 |
| 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 |
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|--|--------|
| 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 |
| 01376 | EBCDIC, DBCS | Hong Kong T-Chinese Host enhancement for HKSCS | T6 |
| 01377 | EBCDIC, MBCS | Hong Kong T-Chinese Mixed Host enhancement for HKSCS | T5 |
| 01378 | EBCDIC, DBCS | T-CH Extended Host DBCS | U4 |
| 01379 | EBCDIC, MBCS | T-CH Extended Mixed Host, Growing | U5 |
| 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 |
| 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 |
| 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 |
| 04393 | EBCDIC, SBCS | FRANCE | BO |
| 04396 | EBCDIC, DBCS | JAPAN | BR |
| 04397 | ASCII, DBCS | JAPAN | BW |
| 04516 | EBCDIC, SBCS | ARABIC | B2 |

| <i>Table 59. CCSIDs supported by z/OS Unicode (continued)</i> | | | |
|---|------------------------|------------------------------|---------------|
| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
| 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 |
| 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 |
| 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 |
| 04970 | ASCII, SBCS | THAI | G4 |
| 04971 | EBCDIC, SBCS | GREEK | G9 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|---|--------|
| 04976 | EBCDIC, SBCS | CYRILLIC | HC |
| 04992 | ASCII, SBCS | JAPANESE TCP-2022 | HJ |
| 04993 | ASCII, SBCS | JAPAN | HL |
| 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 | IO |
| 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 |
| 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 |
| 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 |
| 05233 | EBCDIC, SBCS | DEVANAGARI EBCDIC with Rupee | UO |
| 05255 | EBCDIC, SBCS | T-CHINESE EBCDIC | U6 |
| 05346 | ASCII, SBCS | MS-WIN LATIN-2 | PP |
| 05347 | ASCII, SBCS | MS-WIN CYRILLIC | PR |

| <i>Table 59. CCSIDs supported by z/OS Unicode (continued)</i> | | | |
|---|------------------------|---|---------------|
| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
| 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 |
| 05470 | ASCII, DBCS | BIG-5 EXTENSION FOR HKSCS 2001 | T2 |
| 05471 | ASCII, 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 |
| 05474 | EBCDIC, DBCS | T-CH Extended DBCS Host, Fixed | U2 |
| 05475 | EBCDIC, MBCS | T-CH Mixed Extended, Host, Fixed CS | U3 |
| 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 |
| 05479 | ASCII, MBCS | SIMPLIFIED CHINESE (S-CH) EUC | Q3 |
| 05486 | EBCDIC, MBCS | JAPAN MIXED EBCDIC | UM |
| 05487 | ASCII, QBCS | SIMPLIFIED CHINESE (S-CH)- FOR GB 18030 | TC |
| 05488 | ASCII, MBCS | SIMPLIFIED CHINESE (S-CH) - GB18030 | TB |
| 05495 | EBCDIC, MBCS | JAPAN MIXED EBCDIC | UN |
| 08229 | EBCDIC, SBCS | INTERNATIONAL | AC |
| 08448 | EBCDIC, SBCS | INTERNATIONAL | AK |
| 08482 | EBCDIC, SBCS | JAPAN | BJ |
| 08492 | EBCDIC, DBCS | JAPAN | BS |
| 08493 | ASCII, DBCS | JAPAN HP15-J (DEFINED BY HEWLETT PACKARD) | BX |
| 08612 | EBCDIC, SBCS | ARABIC | B3 |
| 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 |
| 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 |
| 09049 | ASCII, SBCS | TURKISH | FE |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|---|--------|
| 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 |
| 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 |
| 09163 | ASCII, DBCS | KOREAN EUC, G1 | S2 |
| 09219 | EBCDIC, SBCS | JAPAN LATIN EBCDIC | 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 |
| 09563 | EBCDIC, MBCS | T-CHINESE MIX EBC | U7 |
| 09566 | ASCII, DBCS | Big-5 extension for HKSCS-2004, fixed CS, DBCS portion | US |
| 09567 | ASCII, MBCS | Big-5 extension for HKSCS-2004, fixed character set | UT |
| 09568 | EBCDIC, DBCS | Hong Kong T-Chinese Host DB enhancement for HKSCS-2004 | UU |
| 09569 | EBCDIC, MBCS | Hong Kong T-Chinese Mixed Host enhancement for HKSCS-2004 | UV |
| 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 |
| 12544 | EBCDIC, SBCS | FRANCE | AL |
| 12578 | EBCDIC, SBCS | JAPAN EBCDIC SB | SW |
| 12588 | EBCDIC, DBCS | JAPAN | BT |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|---|--------|
| 12712 | EBCDIC, SBCS | HEBREW | CD |
| 12725 | ASCII, SBCS | FRANCE | CH |
| 12788 | EBCDIC, SBCS | ITALY | CU |
| 13121 | EBCDIC, SBCS | KOREA | DL |
| 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 |
| 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 |
| 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 |
| 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 |
| 13488 | UTF-16 | UCS-2 (VERSION 2.0) | PG |
| 13662 | ASCII, DBCS | Big-5 extension for HKSCS-2008, fixed CS, DBCS portion | UX |
| 13663 | ASCII, MBCS | Big-5 extension for HKSCS-2008, fixed CS, MBCS | UZ |
| 13664 | EBCDIC, DBCS | Hong Kong T-Chinese Host DB enhancement for HKSCS-2008 | UW |
| 13665 | EBCDIC, MBCS | Hong Kong T-Chinese Mixed Host enhancement for HKSCS-2008 | UY |
| 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 |
| 17219 | EBCDIC, DBCS | T-CHINESE DB EBCD | U8 |
| 17221 | EBCDIC, DBCS | SIMPLIFIED CHINESE (S-CH)-HOST FOR GBK | TL |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|--------------------------------------|--------|
| 17240 | ASCII, SBCS | HEBREW (WITH MS CONTROLS) | OX |
| 17248 | ASCII, SBCS | ARABIC | F2 |
| 17314 | EBCDIC, MBCS | JAPAN | IU |
| 17331 | ASCII, DBCS | TRADITIONAL CHINESE (T-CH)-IBM BIG-5 | KD |
| 17337 | ASCII, DBCS | JAPANESE TCP-2022 G3-JIS | S4 |
| 17338 | ASCII, MBCS | JAPANESE EUC | UI |
| 17354 | ASCII, MBCS | KOREAN TCP | LQ |
| 17584 | UTF-16 | UCS-2 (VERSION 3.0) | 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 |
| 21434 | ASCII, MBCS | JAPANESE EUC | UJ |
| 21450 | ASCII, MBCS | KOREAN TCP | LR |
| 21680 | UTF-16 | UCS-2 (VERSION 4.0) | TH |
| 21868 | EBCDIC, MBCS | S-Ch Host mixed for GBK - 2015 | U9 |
| 24613 | EBCDIC, SBCS | INTERNATIONAL | AG |
| 24876 | EBCDIC, DBCS | JAPAN | UG |
| 24877 | ASCII, DBCS | JAPAN PC-DISPLAY | BY |
| 25013 | ASCII, SBCS | USA PC-DISPLAY | CK |
| 25076 | EBCDIC, SBCS | DENMARK, NORWAY | CX |
| 25426 | ASCII, SBCS | LATIN-1 PC-DISPLAY | ED |
| 25427 | ASCII, SBCS | GREECE PC-DISPLAY | EI |
| 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 |
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|---------------------------------------|--------|
| 25442 | ASCII, SBCS | CYRILLIC PC-DISPLAY | GE |
| 25444 | ASCII, SBCS | URDU PC-DISPLAY | GK |
| 25445 | ASCII, SBCS | GREECE PC-DISPLAY | GS |
| 25450 | ASCII, SBCS | THAILAND PC-DISPLAY | G6 |
| 25467 | ASCII, SBCS | KOREA PC-DISPLAY | HE |
| 25473 | ASCII, SBCS | JAPAN PC-DISPLAY | HO |
| 25479 | ASCII, SBCS | SIMPLIFIED CHINESE (S-CH) PC-DISPLAY | HX |
| 25480 | ASCII, SBCS | TRADITIONAL CHINESE (T-CH) PC-DISPLAY | HZ |
| 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 |
| 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 |
| 25528 | ASCII, DBCS | JAPANESE EUC | UK |
| 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 |
| 25690 | ASCII, SBCS | TRADITIONAL CHINESE (T-CH)PC-DISPLAY | NK |
| 25691 | ASCII, SBCS | SIMPLIFIED CHINESE (S-CH) GB | NO |
| 25964 | EBCDIC, MBCS | S-Ch Host mixed for GBK - 2023 | VF |
| 28709 | EBCDIC, SBCS | TRADITIONAL CHINESE (T-CH) | AH |
| 29109 | ASCII, SBCS | USA PC-DISPLAY | CL |
| 29172 | EBCDIC, SBCS | BRAZIL | CY |
| 29509 | EBCDIC, DBCS | S-CHINESE EBCDIC DB | U1 |
| 29522 | ASCII, SBCS | LATIN-1 PC-DISPLAY | EE |
| 29523 | ASCII, SBCS | GREECE PC-DISPLAY | EJ |
| 29524 | ASCII, SBCS | LATIN-2 PC-DISPLAY | EP |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|---------------------------------------|--------|
| 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 |
| 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 |
| 32805 | EBCDIC, SBCS | JAPAN LATIN | AI |
| 33058 | EBCDIC, SBCS | JAPAN | BK |
| 33605 | EBCDIC, DBCS | S-CHINESE EBCDIC DB | VE |
| 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 |
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| 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 |
| 37719 | ASCII, SBCS | CYRILLIC PC-DISPLAY | E3 |
| 37728 | ASCII, SBCS | ARABIC PC-DISPLAY | F6 |
| 37732 | ASCII, SBCS | URDU PC-DISPLAY | GN |
| 37761 | ASCII, SBCS | JAPAN PC-DISPLAY | HQ |
| 37796 | ASCII, MBCS | JAPAN PC-DISPLAY | I4 |
| 37813 | ASCII, MBCS | KOREA KS PC-DISPLAY | KN |
| 37818 | ASCII, MBCS | JAPANESE EUC | UL |
| 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 |
| 42160 | UTF-16 | Unicode 6.0, UTF-16 BE with IBM PUA | UR |
| 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 |
| 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 |
| 54448 | UTF-16 | Unicode 9.0, UTF-16 BE with IBM PUA | VA |
| 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 |

Table 59. CCSIDs supported by z/OS Unicode (continued)

| CCSID | ENCODING SCHEME | DESCRIPTION | SUFFIX |
|-------|-----------------|---|--------|
| 61712 | EBCDIC, SBCS | GLOBAL USE | AU |
| 61956 | UTF-16 | WITH MAPPING OF PUA CHARACTERS AS PRESCRIBED BY MICROSOFT | T0 |
| 62273 | EBCDIC, SBCS | Special - KOREAN EBCDIC | UQ |
| 62337 | ASCII, SBCS | Special - JAPAN SB PC-DATA | UD |
| 62373 | EBCDIC, MBCS | Special - KOREAN MIXED EBCDIC | UP |
| 62381 | ASCII, DBCS | Special - JAPAN DB PC-DATA | UE |
| 62383 | ASCII, MBCS | Special - JAPAN OPEN | UC |

Appendix B. Conversion support for multi-byte encodings (MBCS)

This topic describes how z/OS Unicode Services converts data when the data is tagged with MBCS (multibyte character set) CCSIDs.

Internal handling of MBCS conversions

Whenever an MBCS CCSID is specified for a conversion, z/OS support for Unicode decomposes the MBCS CCSID into its sub CCSIDs (the SBCS and DBCS parts) and then uses the conversion tables for each part. There are no direct Unicode to MBCS tables provided.

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.

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 | | |

| MBCS | Sub 1 | Sub2 | Sub3 | Sub4 | Sub5 | Sub6 |
|-------------|--------------|-------------|-------------|-------------|-------------|-------------|
| 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 | | | |
| 00970 | 00367 | 00971 | | | | |
| 01350 | 00367 | 05048 | 13184 | 05049 | | |
| 01363 | 01126 | 01362 | | | | |
| 01364 | 13121 | 04930 | | | | |
| 01370 | 05210 | 21427 | | | | |
| 01371 | 01159 | 09027 | | | | |
| 01375 | 09444 | 01374 | | | | |
| 01379 | 28709 | 01378 | | | | |
| 01381 | 01115 | 01380 | | | | |
| 01383 | 00367 | 01382 | | | | |
| 01386 | 05210 | 01385 | | | | |
| 01388 | 13124 | 04933 | | | | |
| 01390 | 08482 | 16684 | | | | |
| 01392 | 09444 | 09577 | 01391 | | | |
| 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 | | |

| MBCS | Sub 1 | Sub2 | Sub3 | Sub4 | Sub5 | Sub6 |
|-------------|--------------|-------------|-------------|-------------|-------------|-------------|
| 05055 | 00367 | 00955 | 00896 | 21433 | | |
| 05471 | 09444 | 05470 | | | | |
| 05473 | 28709 | 05472 | | | | |
| 05475 | 28709 | 05474 | | | | |
| 05477 | 05211 | 01380 | | | | |
| 05479 | 00367 | 09574 | | | | |
| 05486 | 12578 | 20780 | | | | |
| 05488 | 09444 | 09577 | 05487 | | | |
| 05495 | 09219 | 20780 | | | | |
| 09122 | 04386 | 00300 | | | | |
| 09124 | 09089 | 00301 | | | | |
| 09125 | 09025 | 09026 | | | | |
| 09127 | 09028 | 00837 | | | | |
| 09131 | 01027 | 08493 | | | | |
| 09142 | 01114 | 09139 | | | | |
| 09146 | 00895 | 00952 | 13184 | 00953 | | |
| 09575 | 00367 | 05478 | | | | |
| 09580 | 00836 | 13125 | | | | |
| 13218 | 04386 | 04396 | | | | |
| 13219 | 08229 | 04396 | | | | |
| 13238 | 01114 | 13235 | | | | |
| 13242 | 00895 | 05048 | 13184 | 05049 | | |
| 13676 | 00836 | 17221 | | | | |
| 17314 | 00290 | 12588 | | | | |
| 17354 | 00367 | 09163 | | | | |
| 17338 | 00895 | 25528 | 13184 | 09145 | | |
| 21434 | 00895 | 25528 | 09088 | 09145 | | |
| 21450 | 00367 | 05067 | | | | |
| 25508 | 25473 | 24877 | | | | |
| 25510 | 25467 | 25502 | | | | |
| 25512 | 25479 | 25504 | | | | |
| 25514 | 25480 | 25503 | | | | |
| 25518 | 25617 | 24877 | | | | |
| 25520 | 25616 | 25502 | | | | |
| 25522 | 25618 | 25504 | | | | |

| MBCS | Sub 1 | Sub2 | Sub3 | Sub4 | Sub5 | Sub6 |
|-------|-------|-------|-------|-------|------|------|
| 25524 | 25619 | 25503 | | | | |
| 25525 | 25664 | 25527 | | | | |
| 25546 | 00367 | 09163 | | | | |
| 25964 | 836 | 33605 | | | | |
| 29614 | 29713 | 24877 | | | | |
| 29616 | 29712 | 25502 | | | | |
| 29618 | 29714 | 25504 | | | | |
| 29620 | 29715 | 25503 | | | | |
| 29621 | 29760 | 25527 | | | | |
| 33698 | 33058 | 04396 | | | | |
| 33699 | 32805 | 04396 | | | | |
| 33700 | 33665 | 24877 | | | | |
| 33717 | 25664 | 29623 | | | | |
| 33722 | 00895 | 00952 | 09088 | 09145 | | |
| 37796 | 37761 | 24877 | | | | |
| 37813 | 29760 | 29623 | | | | |
| 37818 | 00895 | 25528 | 09088 | 00953 | | |

When a shift character is in the data stream

In this 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 332 illustrates this method.

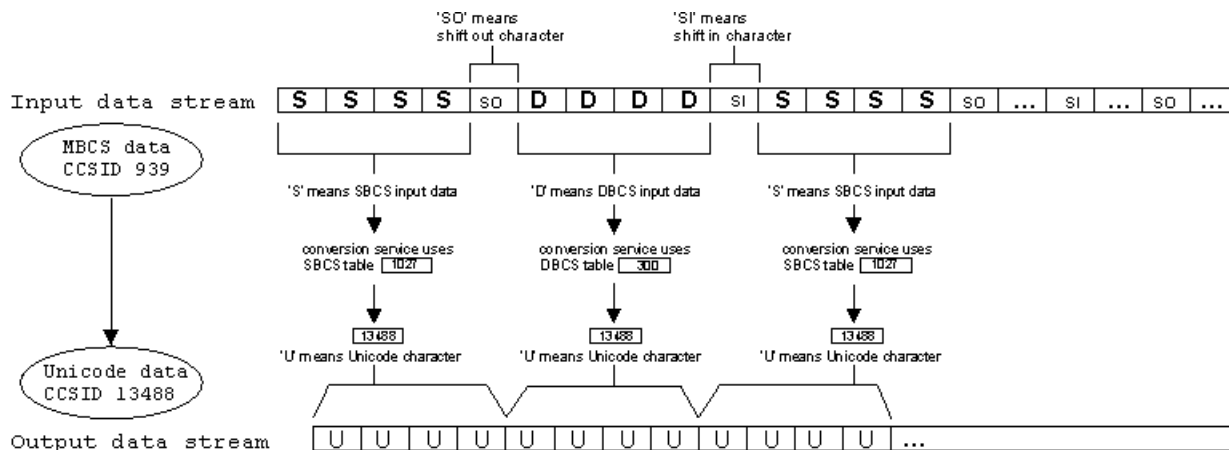


Figure 1. Conversion of MBCS data to Unicode characters

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 332 the 'shift out' character is indicated by **SO** and the 'shift in' character by **SI**).

The converter 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.

MBCS CCSIDs compatible with iconv

The following is a list of MBCS CCSID tables that were changed to provide compatibility with the C Run-time 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 conversion tables described in Appendix C, "Conversion tables supplied with z/OS Unicode Services," on page 335 that support technique L can be used. Technique L is described in ["Creating a conversion image"](#) on page 254.

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 C Run-time 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
33722 corresponds to IBM-eucJC

Appendix C. Conversion tables supplied with z/OS Unicode Services

The following CCSID conversions types are supported for direct conversions:

Table 60. CCSID conversions types of z/OS support for the Unicode Standard

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

Direct conversions supported between non-Unicode CCSIDs

The following table lists the techniques supported as direct conversions between non-Unicode CCSIDs.

Table 61. Non-Unicode Conversions Available

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00037 | 00256 | R,E |
| 00037 | 00273 | R |
| 00037 | 00275 | R |
| 00037 | 00277 | R,E |
| 00037 | 00278 | R,E |
| 00037 | 00280 | R,E |
| 00037 | 00284 | R,E |
| 00037 | 00285 | R,E |
| 00037 | 00290 | R,E |
| 00037 | 00297 | R,E |
| 00037 | 00367 | E |
| 00037 | 00420 | R,E |
| 00037 | 00423 | R,E |
| 00037 | 00424 | R,E |
| 00037 | 00425 | R,E |
| 00037 | 00437 | R,E,M,P |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00037 | 00500 | R,E |
| 00037 | 00720 | R |
| 00037 | 00737 | R |
| 00037 | 00775 | R |
| 00037 | 00813 | R,L |
| 00037 | 00819 | R,L |
| 00037 | 00833 | R,E |
| 00037 | 00836 | R,E |
| 00037 | 00838 | E |
| 00037 | 00850 | R,E,M |
| 00037 | 00852 | R,E,M |
| 00037 | 00855 | R,M |
| 00037 | 00857 | R,E |
| 00037 | 00858 | R,E,M |
| 00037 | 00860 | R,E |
| 00037 | 00861 | R,E,M |
| 00037 | 00862 | R,E,M |
| 00037 | 00863 | R,E |
| 00037 | 00864 | R,E,M |
| 00037 | 00865 | R,E |
| 00037 | 00866 | R,M |
| 00037 | 00869 | R,M |
| 00037 | 00870 | R,E |
| 00037 | 00871 | R,E |
| 00037 | 00874 | R,E,M |
| 00037 | 00875 | R,E |
| 00037 | 00880 | R,E |
| 00037 | 00897 | R,E |
| 00037 | 00901 | R,E,L |
| 00037 | 00902 | R,E,L |
| 00037 | 00903 | R |
| 00037 | 00904 | E,L |
| 00037 | 00905 | R,E |
| 00037 | 00912 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00037 | 00914 | R,L |
| 00037 | 00915 | R,L |
| 00037 | 00916 | R,L |
| 00037 | 00920 | R,L |
| 00037 | 00921 | R,L |
| 00037 | 00922 | R,L |
| 00037 | 00923 | R,E,L |
| 00037 | 00924 | R,E |
| 00037 | 01009 | E |
| 00037 | 01025 | R,E |
| 00037 | 01026 | R,E |
| 00037 | 01027 | R,E |
| 00037 | 01040 | R,E |
| 00037 | 01041 | R,E |
| 00037 | 01042 | R |
| 00037 | 01043 | R,E |
| 00037 | 01047 | R |
| 00037 | 01051 | R,E |
| 00037 | 01088 | R,L |
| 00037 | 01089 | R,E,L |
| 00037 | 01097 | R,E |
| 00037 | 01100 | R |
| 00037 | 01112 | R,E |
| 00037 | 01114 | E |
| 00037 | 01115 | E,L |
| 00037 | 01122 | R |
| 00037 | 01123 | R,E |
| 00037 | 01124 | R,E,L |
| 00037 | 01126 | E,L |
| 00037 | 01130 | R |
| 00037 | 01131 | R,E |
| 00037 | 01132 | R |
| 00037 | 01137 | E |
| 00037 | 01140 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00037 | 01141 | R,E |
| 00037 | 01142 | R,E |
| 00037 | 01143 | R,E |
| 00037 | 01144 | R,E |
| 00037 | 01145 | R,E |
| 00037 | 01146 | R,E |
| 00037 | 01147 | R,E |
| 00037 | 01148 | R,E |
| 00037 | 01149 | R,E |
| 00037 | 01250 | R,L |
| 00037 | 01251 | R,L |
| 00037 | 01252 | R,E,L |
| 00037 | 01253 | R,L |
| 00037 | 01254 | R,L |
| 00037 | 01255 | R,L |
| 00037 | 01257 | R |
| 00037 | 01258 | R,E |
| 00037 | 01275 | R |
| 00037 | 01280 | R |
| 00037 | 01281 | R |
| 00037 | 01283 | R |
| 00037 | 04909 | R,E,L |
| 00037 | 05210 | E |
| 00037 | 05348 | R,E,L |
| 00256 | 00037 | R,E |
| 00256 | 00273 | R |
| 00256 | 00277 | R |
| 00256 | 00278 | R |
| 00256 | 00280 | R |
| 00256 | 00284 | R |
| 00256 | 00285 | R |
| 00256 | 00290 | E |
| 00256 | 00297 | R |
| 00256 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00256 | 00420 | R |
| 00256 | 00423 | R |
| 00256 | 00424 | R |
| 00256 | 00437 | R,E |
| 00256 | 00500 | R,E |
| 00256 | 00737 | R |
| 00256 | 00775 | R,E |
| 00256 | 00819 | R |
| 00256 | 00833 | E |
| 00256 | 00836 | E |
| 00256 | 00838 | E |
| 00256 | 00850 | R,E |
| 00256 | 00852 | R,E |
| 00256 | 00857 | R,E |
| 00256 | 00860 | R,E |
| 00256 | 00861 | R,E |
| 00256 | 00862 | R,E |
| 00256 | 00863 | R,E |
| 00256 | 00864 | R,E |
| 00256 | 00865 | R,E |
| 00256 | 00866 | E,C |
| 00256 | 00869 | R |
| 00256 | 00870 | R,E |
| 00256 | 00871 | R |
| 00256 | 00875 | R |
| 00256 | 00880 | R |
| 00256 | 00905 | R |
| 00256 | 01025 | R |
| 00256 | 01026 | R |
| 00256 | 01027 | E |
| 00256 | 01112 | R |
| 00256 | 01122 | R |
| 00256 | 01251 | R,E |
| 00256 | 01252 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00256 | 01275 | R |
| 00259 | 00437 | E |
| 00259 | 00808 | E |
| 00259 | 00850 | E |
| 00259 | 00851 | E |
| 00259 | 00852 | E |
| 00259 | 00855 | R,E |
| 00259 | 00856 | E |
| 00259 | 00857 | E |
| 00259 | 00858 | E |
| 00259 | 00860 | E |
| 00259 | 00861 | E |
| 00259 | 00862 | E |
| 00259 | 00863 | E |
| 00259 | 00864 | E |
| 00259 | 00865 | E |
| 00259 | 00866 | E |
| 00259 | 00867 | E |
| 00259 | 00869 | E |
| 00259 | 00872 | E |
| 00259 | 00874 | E |
| 00259 | 00899 | E |
| 00259 | 00901 | E |
| 00259 | 00902 | E |
| 00259 | 00915 | R,E |
| 00259 | 01051 | E |
| 00259 | 01098 | R,E |
| 00259 | 01161 | E |
| 00259 | 01162 | E |
| 00259 | 01250 | E |
| 00259 | 01251 | E |
| 00259 | 01252 | E |
| 00259 | 01253 | E |
| 00259 | 01254 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00259 | 01255 | E |
| 00259 | 01256 | E |
| 00259 | 01257 | E |
| 00259 | 01258 | E |
| 00259 | 05348 | E |
| 00273 | 00037 | R,E |
| 00273 | 00256 | R |
| 00273 | 00277 | R |
| 00273 | 00278 | R |
| 00273 | 00280 | R |
| 00273 | 00284 | R |
| 00273 | 00285 | R |
| 00273 | 00290 | R,E |
| 00273 | 00297 | R |
| 00273 | 00367 | E |
| 00273 | 00423 | R |
| 00273 | 00437 | R,E,M,P |
| 00273 | 00500 | R,E |
| 00273 | 00737 | R |
| 00273 | 00775 | R |
| 00273 | 00813 | R,L |
| 00273 | 00819 | R,L |
| 00273 | 00833 | R,E |
| 00273 | 00836 | R,E |
| 00273 | 00838 | E |
| 00273 | 00850 | R,E,M |
| 00273 | 00852 | R,E,M |
| 00273 | 00855 | R,M |
| 00273 | 00856 | E,L |
| 00273 | 00857 | R,E |
| 00273 | 00858 | R,E,M |
| 00273 | 00860 | R,E |
| 00273 | 00861 | R,E,M |
| 00273 | 00862 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00273 | 00863 | R,E |
| 00273 | 00864 | R,E,M |
| 00273 | 00865 | R,E |
| 00273 | 00869 | R,M |
| 00273 | 00870 | R |
| 00273 | 00871 | R |
| 00273 | 00874 | R,M |
| 00273 | 00875 | R |
| 00273 | 00880 | R |
| 00273 | 00897 | R |
| 00273 | 00903 | R |
| 00273 | 00912 | R,L |
| 00273 | 00916 | R,L |
| 00273 | 00920 | R,L |
| 00273 | 00923 | R,E,L |
| 00273 | 00924 | R,E |
| 00273 | 01009 | E |
| 00273 | 01025 | R |
| 00273 | 01026 | R |
| 00273 | 01027 | R,E |
| 00273 | 01040 | R,E |
| 00273 | 01041 | R,E |
| 00273 | 01042 | R |
| 00273 | 01043 | R,E |
| 00273 | 01047 | R |
| 00273 | 01051 | R,E |
| 00273 | 01088 | R,L |
| 00273 | 01100 | R |
| 00273 | 01112 | R |
| 00273 | 01122 | R |
| 00273 | 01140 | R,E |
| 00273 | 01141 | E |
| 00273 | 01142 | R,E |
| 00273 | 01143 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00273 | 01144 | R,E |
| 00273 | 01145 | R,E |
| 00273 | 01146 | R,E |
| 00273 | 01147 | R,E |
| 00273 | 01148 | R,E |
| 00273 | 01149 | R,E |
| 00273 | 01250 | R,E,L |
| 00273 | 01252 | R,E,L |
| 00273 | 01275 | R |
| 00273 | 05348 | R,E,L |
| 00274 | 00500 | R |
| 00274 | 00819 | R,E |
| 00274 | 00850 | R,E,M |
| 00274 | 01047 | R |
| 00274 | 01148 | R,E |
| 00274 | 01252 | R,E,L |
| 00275 | 00037 | R |
| 00275 | 00437 | R,E,M |
| 00275 | 00500 | R |
| 00275 | 00819 | R,E |
| 00275 | 00850 | R,E,M |
| 00275 | 01047 | R |
| 00275 | 01148 | R,E |
| 00275 | 01252 | R,E,L |
| 00275 | 05348 | R,E,L |
| 00277 | 00037 | R,E |
| 00277 | 00256 | R |
| 00277 | 00273 | R |
| 00277 | 00278 | R |
| 00277 | 00280 | R |
| 00277 | 00284 | R |
| 00277 | 00285 | R |
| 00277 | 00290 | R,E |
| 00277 | 00297 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00277 | 00367 | E |
| 00277 | 00423 | R |
| 00277 | 00437 | R,E,M,P |
| 00277 | 00500 | R,E |
| 00277 | 00737 | R |
| 00277 | 00775 | R,E |
| 00277 | 00813 | R,L |
| 00277 | 00819 | R,L |
| 00277 | 00833 | R,E |
| 00277 | 00836 | R,E |
| 00277 | 00838 | E |
| 00277 | 00850 | R,E,M |
| 00277 | 00852 | R,E,M |
| 00277 | 00855 | R,M |
| 00277 | 00857 | R,E |
| 00277 | 00858 | R,E,M |
| 00277 | 00860 | R,E |
| 00277 | 00861 | R,E,M |
| 00277 | 00862 | R,E,M |
| 00277 | 00863 | R,E |
| 00277 | 00864 | R,E,M |
| 00277 | 00865 | R,E |
| 00277 | 00869 | R,M |
| 00277 | 00870 | R |
| 00277 | 00871 | R |
| 00277 | 00874 | R,M |
| 00277 | 00875 | R |
| 00277 | 00880 | R |
| 00277 | 00897 | R |
| 00277 | 00903 | R |
| 00277 | 00912 | R,L |
| 00277 | 00916 | R,L |
| 00277 | 00920 | R,L |
| 00277 | 00923 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00277 | 00924 | R,E |
| 00277 | 01009 | E |
| 00277 | 01025 | R |
| 00277 | 01026 | R |
| 00277 | 01027 | R,E |
| 00277 | 01040 | R,E |
| 00277 | 01041 | R,E |
| 00277 | 01042 | R |
| 00277 | 01043 | R,E |
| 00277 | 01047 | R |
| 00277 | 01051 | R,E |
| 00277 | 01088 | R,L |
| 00277 | 01100 | R |
| 00277 | 01112 | R |
| 00277 | 01122 | R |
| 00277 | 01140 | R,E |
| 00277 | 01141 | R,E |
| 00277 | 01142 | E |
| 00277 | 01143 | R,E |
| 00277 | 01144 | R,E |
| 00277 | 01145 | R,E |
| 00277 | 01146 | R,E |
| 00277 | 01147 | R,E |
| 00277 | 01148 | R,E |
| 00277 | 01149 | R,E |
| 00277 | 01252 | R,E,L |
| 00277 | 01275 | R |
| 00277 | 05348 | R,E,L |
| 00278 | 00037 | R,E |
| 00278 | 00256 | R |
| 00278 | 00273 | R |
| 00278 | 00277 | R |
| 00278 | 00280 | R |
| 00278 | 00284 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00278 | 00285 | R |
| 00278 | 00290 | R,E |
| 00278 | 00297 | R |
| 00278 | 00367 | E |
| 00278 | 00423 | R |
| 00278 | 00437 | R,E,M,P |
| 00278 | 00500 | R,E |
| 00278 | 00737 | R |
| 00278 | 00775 | R |
| 00278 | 00813 | R,L |
| 00278 | 00819 | R,L |
| 00278 | 00833 | R,E |
| 00278 | 00836 | R,E |
| 00278 | 00838 | E |
| 00278 | 00850 | R,E,M |
| 00278 | 00852 | R,E,M |
| 00278 | 00855 | R,M |
| 00278 | 00857 | R,E |
| 00278 | 00858 | R,E,M |
| 00278 | 00860 | R,E |
| 00278 | 00861 | R,E,M |
| 00278 | 00862 | R,E,M |
| 00278 | 00863 | R,E |
| 00278 | 00864 | R,E,M |
| 00278 | 00865 | R,E |
| 00278 | 00869 | R,M |
| 00278 | 00870 | R |
| 00278 | 00871 | R |
| 00278 | 00874 | R,M |
| 00278 | 00875 | R |
| 00278 | 00880 | R |
| 00278 | 00897 | R |
| 00278 | 00903 | R |
| 00278 | 00912 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00278 | 00916 | R,L |
| 00278 | 00920 | R,L |
| 00278 | 00923 | R,E,L |
| 00278 | 00924 | R,E |
| 00278 | 01009 | E |
| 00278 | 01025 | R |
| 00278 | 01026 | R |
| 00278 | 01027 | R,E |
| 00278 | 01040 | R,E |
| 00278 | 01041 | R,E |
| 00278 | 01042 | R |
| 00278 | 01043 | R,E |
| 00278 | 01047 | R |
| 00278 | 01051 | R,E |
| 00278 | 01088 | R,L |
| 00278 | 01100 | R |
| 00278 | 01112 | R |
| 00278 | 01122 | R |
| 00278 | 01140 | R,E |
| 00278 | 01141 | R,E |
| 00278 | 01142 | R,E |
| 00278 | 01143 | E |
| 00278 | 01144 | R,E |
| 00278 | 01145 | R,E |
| 00278 | 01146 | R,E |
| 00278 | 01147 | R,E |
| 00278 | 01148 | R,E |
| 00278 | 01149 | R,E |
| 00278 | 01252 | R,E,L |
| 00278 | 01275 | R |
| 00278 | 05348 | R,E,L |
| 00280 | 00037 | R,E |
| 00280 | 00256 | R |
| 00280 | 00273 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00280 | 00277 | R |
| 00280 | 00278 | R |
| 00280 | 00284 | R |
| 00280 | 00285 | R |
| 00280 | 00290 | R,E |
| 00280 | 00297 | R |
| 00280 | 00367 | E |
| 00280 | 00423 | R |
| 00280 | 00437 | R,E,M,P |
| 00280 | 00500 | R,E |
| 00280 | 00737 | R |
| 00280 | 00775 | R,E |
| 00280 | 00813 | R,L |
| 00280 | 00819 | R,L |
| 00280 | 00833 | R,E |
| 00280 | 00836 | R,E |
| 00280 | 00838 | E |
| 00280 | 00850 | R,E,M |
| 00280 | 00852 | R,E,M |
| 00280 | 00855 | R,M |
| 00280 | 00857 | R,E |
| 00280 | 00858 | R,E,M |
| 00280 | 00860 | R,E |
| 00280 | 00861 | R,E,M |
| 00280 | 00862 | R,E,M |
| 00280 | 00863 | R,E |
| 00280 | 00864 | R,E,M |
| 00280 | 00865 | R,E |
| 00280 | 00869 | R,M |
| 00280 | 00870 | R |
| 00280 | 00871 | R |
| 00280 | 00874 | R,M |
| 00280 | 00875 | R |
| 00280 | 00880 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00280 | 00897 | R |
| 00280 | 00903 | R |
| 00280 | 00912 | R,L |
| 00280 | 00916 | R,L |
| 00280 | 00920 | R,L |
| 00280 | 00923 | R,E,L |
| 00280 | 00924 | R,E |
| 00280 | 01009 | E |
| 00280 | 01025 | R |
| 00280 | 01026 | R |
| 00280 | 01027 | R,E |
| 00280 | 01040 | R,E |
| 00280 | 01041 | R,E |
| 00280 | 01042 | R |
| 00280 | 01043 | R,E |
| 00280 | 01047 | R |
| 00280 | 01051 | R,E |
| 00280 | 01088 | R,L |
| 00280 | 01100 | R |
| 00280 | 01112 | R |
| 00280 | 01122 | R |
| 00280 | 01140 | R,E |
| 00280 | 01141 | R,E |
| 00280 | 01142 | R,E |
| 00280 | 01143 | R,E |
| 00280 | 01144 | E |
| 00280 | 01145 | R,E |
| 00280 | 01146 | R,E |
| 00280 | 01147 | R,E |
| 00280 | 01148 | R,E |
| 00280 | 01149 | R,E |
| 00280 | 01252 | R,E,L |
| 00280 | 01275 | R |
| 00280 | 05348 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00281 | 00500 | R,E |
| 00281 | 00819 | R,E,L |
| 00281 | 01047 | R |
| 00281 | 01148 | R,E |
| 00282 | 00500 | R |
| 00282 | 00819 | R,E,L |
| 00282 | 01047 | R |
| 00282 | 01051 | E |
| 00282 | 01148 | R,E |
| 00284 | 00037 | R,E |
| 00284 | 00256 | R |
| 00284 | 00273 | R |
| 00284 | 00277 | R |
| 00284 | 00278 | R |
| 00284 | 00280 | R |
| 00284 | 00285 | R |
| 00284 | 00290 | R,E |
| 00284 | 00297 | R |
| 00284 | 00367 | E |
| 00284 | 00423 | R |
| 00284 | 00437 | R,E,M,P |
| 00284 | 00500 | R,E |
| 00284 | 00737 | R |
| 00284 | 00775 | R |
| 00284 | 00813 | R,L |
| 00284 | 00819 | R,L |
| 00284 | 00833 | R,E |
| 00284 | 00836 | R,E |
| 00284 | 00838 | E |
| 00284 | 00850 | R,E,M |
| 00284 | 00852 | R,E,M |
| 00284 | 00855 | R,M |
| 00284 | 00857 | R,E |
| 00284 | 00858 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00284 | 00860 | R,E |
| 00284 | 00861 | R,E,M |
| 00284 | 00862 | R,E,M |
| 00284 | 00863 | R,E |
| 00284 | 00864 | R,E,M |
| 00284 | 00865 | R,E |
| 00284 | 00869 | R,M |
| 00284 | 00870 | R |
| 00284 | 00871 | R |
| 00284 | 00874 | R,M |
| 00284 | 00875 | R |
| 00284 | 00880 | R |
| 00284 | 00897 | R |
| 00284 | 00903 | R |
| 00284 | 00912 | R,L |
| 00284 | 00916 | R,L |
| 00284 | 00920 | R,L |
| 00284 | 00923 | R,E,L |
| 00284 | 00924 | R,E |
| 00284 | 01009 | E |
| 00284 | 01025 | R |
| 00284 | 01026 | R |
| 00284 | 01027 | R,E |
| 00284 | 01040 | R,E |
| 00284 | 01041 | R,E |
| 00284 | 01042 | R |
| 00284 | 01043 | R,E |
| 00284 | 01047 | R |
| 00284 | 01051 | R,E |
| 00284 | 01088 | R,L |
| 00284 | 01100 | R |
| 00284 | 01112 | R |
| 00284 | 01122 | R |
| 00284 | 01140 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00284 | 01141 | R,E |
| 00284 | 01142 | R,E |
| 00284 | 01143 | R,E |
| 00284 | 01144 | R,E |
| 00284 | 01145 | E |
| 00284 | 01146 | R,E |
| 00284 | 01147 | R,E |
| 00284 | 01148 | R,E |
| 00284 | 01149 | R,E |
| 00284 | 01252 | R,E,L |
| 00284 | 01275 | R |
| 00284 | 05348 | R,E,L |
| 00285 | 00037 | R,E |
| 00285 | 00256 | R |
| 00285 | 00273 | R |
| 00285 | 00277 | R |
| 00285 | 00278 | R |
| 00285 | 00280 | R |
| 00285 | 00284 | R |
| 00285 | 00290 | R,E |
| 00285 | 00297 | R |
| 00285 | 00367 | E |
| 00285 | 00423 | R |
| 00285 | 00437 | R,E,M,P |
| 00285 | 00500 | R,E |
| 00285 | 00737 | R |
| 00285 | 00775 | R,E |
| 00285 | 00813 | R,L |
| 00285 | 00819 | R,L |
| 00285 | 00833 | R,E |
| 00285 | 00836 | R,E |
| 00285 | 00838 | E |
| 00285 | 00850 | R,E,M |
| 00285 | 00852 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00285 | 00855 | R,M |
| 00285 | 00857 | R,E |
| 00285 | 00858 | R,E,M |
| 00285 | 00860 | R,E |
| 00285 | 00861 | R,E,M |
| 00285 | 00862 | R,E,M |
| 00285 | 00863 | R,E |
| 00285 | 00864 | R,E,M |
| 00285 | 00865 | R,E |
| 00285 | 00869 | R,M |
| 00285 | 00870 | R |
| 00285 | 00871 | R |
| 00285 | 00874 | R,M |
| 00285 | 00875 | R |
| 00285 | 00880 | R |
| 00285 | 00897 | R |
| 00285 | 00903 | R |
| 00285 | 00912 | R,L |
| 00285 | 00916 | R,L |
| 00285 | 00920 | R,L |
| 00285 | 00923 | R,E,L |
| 00285 | 00924 | R,E |
| 00285 | 01025 | R |
| 00285 | 01026 | R |
| 00285 | 01027 | R,E |
| 00285 | 01040 | R,E |
| 00285 | 01041 | R,E |
| 00285 | 01042 | R |
| 00285 | 01043 | R,E |
| 00285 | 01047 | R |
| 00285 | 01051 | R,E |
| 00285 | 01088 | R,L |
| 00285 | 01100 | R |
| 00285 | 01112 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00285 | 01122 | R |
| 00285 | 01140 | R,E |
| 00285 | 01141 | R,E |
| 00285 | 01142 | R,E |
| 00285 | 01143 | R,E |
| 00285 | 01144 | R,E |
| 00285 | 01145 | R,E |
| 00285 | 01146 | E |
| 00285 | 01147 | R,E |
| 00285 | 01148 | R,E |
| 00285 | 01149 | R,E |
| 00285 | 01252 | R,E,L |
| 00285 | 01275 | R |
| 00285 | 05348 | R,E,L |
| 00290 | 00037 | R,E |
| 00290 | 00256 | E |
| 00290 | 00273 | R,E |
| 00290 | 00277 | R,E |
| 00290 | 00278 | R,E |
| 00290 | 00280 | R,E |
| 00290 | 00284 | R,E |
| 00290 | 00285 | R,E |
| 00290 | 00297 | R,E |
| 00290 | 00367 | E |
| 00290 | 00437 | R,E,M |
| 00290 | 00500 | R,E |
| 00290 | 00737 | E |
| 00290 | 00775 | E |
| 00290 | 00819 | E,L |
| 00290 | 00833 | R,E |
| 00290 | 00836 | R,E |
| 00290 | 00850 | R,E,M |
| 00290 | 00852 | R,E,M |
| 00290 | 00855 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00290 | 00857 | R,E |
| 00290 | 00858 | E,L |
| 00290 | 00860 | R,E |
| 00290 | 00861 | R,E,M |
| 00290 | 00862 | R,E,M |
| 00290 | 00863 | R,E |
| 00290 | 00864 | R,E,M |
| 00290 | 00865 | R,E |
| 00290 | 00870 | R,E |
| 00290 | 00871 | R,E |
| 00290 | 00895 | E |
| 00290 | 00896 | E |
| 00290 | 00897 | E |
| 00290 | 01009 | E |
| 00290 | 01025 | R,E |
| 00290 | 01026 | R,E |
| 00290 | 01027 | R |
| 00290 | 01040 | R,E |
| 00290 | 01041 | R,E |
| 00290 | 01042 | R |
| 00290 | 01043 | R,E |
| 00290 | 01047 | R,E |
| 00290 | 01088 | R,L |
| 00290 | 01112 | R |
| 00290 | 01122 | R |
| 00290 | 01148 | R,E |
| 00290 | 01252 | E,L |
| 00290 | 05348 | E,L |
| 00297 | 00037 | R,E |
| 00297 | 00256 | R |
| 00297 | 00273 | R |
| 00297 | 00277 | R |
| 00297 | 00278 | R |
| 00297 | 00280 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00297 | 00284 | R |
| 00297 | 00285 | R |
| 00297 | 00290 | R,E |
| 00297 | 00367 | E |
| 00297 | 00423 | R |
| 00297 | 00437 | R,E,M,P |
| 00297 | 00500 | R,E |
| 00297 | 00737 | R |
| 00297 | 00775 | R,E |
| 00297 | 00813 | R,L |
| 00297 | 00819 | R,L |
| 00297 | 00833 | R,E |
| 00297 | 00836 | R,E |
| 00297 | 00838 | E |
| 00297 | 00850 | R,E,M |
| 00297 | 00852 | R,E,M |
| 00297 | 00855 | R,M |
| 00297 | 00857 | R,E |
| 00297 | 00858 | R,E,M |
| 00297 | 00860 | R,E |
| 00297 | 00861 | R,E,M |
| 00297 | 00862 | R,E,M |
| 00297 | 00863 | R,E |
| 00297 | 00864 | R,E,M |
| 00297 | 00865 | R,E |
| 00297 | 00869 | R,M |
| 00297 | 00870 | R |
| 00297 | 00871 | R |
| 00297 | 00874 | R,M |
| 00297 | 00875 | R |
| 00297 | 00880 | R |
| 00297 | 00897 | R |
| 00297 | 00903 | R |
| 00297 | 00912 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00297 | 00916 | R,L |
| 00297 | 00920 | R,L |
| 00297 | 00923 | R,E,L |
| 00297 | 00924 | R,E |
| 00297 | 01009 | E |
| 00297 | 01025 | R |
| 00297 | 01026 | R |
| 00297 | 01027 | R,E |
| 00297 | 01040 | R,E |
| 00297 | 01041 | R,E |
| 00297 | 01042 | R |
| 00297 | 01043 | R,E |
| 00297 | 01047 | R |
| 00297 | 01051 | R,E |
| 00297 | 01088 | R,L |
| 00297 | 01100 | R |
| 00297 | 01112 | R |
| 00297 | 01122 | R |
| 00297 | 01140 | R,E |
| 00297 | 01141 | R,E |
| 00297 | 01142 | R,E |
| 00297 | 01143 | R,E |
| 00297 | 01144 | R,E |
| 00297 | 01145 | R,E |
| 00297 | 01146 | R,E |
| 00297 | 01147 | E |
| 00297 | 01148 | R,E |
| 00297 | 01149 | R,E |
| 00297 | 01252 | R,E,L |
| 00297 | 01275 | R |
| 00297 | 05348 | R,E,L |
| 00300 | 00301 | E |
| 00300 | 00941 | E |
| 00300 | 01351 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00301 | 00300 | E |
| 00301 | 00941 | E |
| 00301 | 01351 | E |
| 00367 | 00037 | E |
| 00367 | 00256 | E |
| 00367 | 00273 | E |
| 00367 | 00277 | E |
| 00367 | 00278 | E |
| 00367 | 00280 | E |
| 00367 | 00284 | E |
| 00367 | 00285 | E |
| 00367 | 00290 | E |
| 00367 | 00297 | E |
| 00367 | 00420 | E |
| 00367 | 00421 | E |
| 00367 | 00423 | E |
| 00367 | 00424 | E |
| 00367 | 00437 | E |
| 00367 | 00500 | E |
| 00367 | 00803 | E |
| 00367 | 00813 | E |
| 00367 | 00819 | E |
| 00367 | 00833 | E |
| 00367 | 00836 | E |
| 00367 | 00838 | E |
| 00367 | 00850 | E |
| 00367 | 00851 | E |
| 00367 | 00852 | E |
| 00367 | 00853 | E |
| 00367 | 00855 | E |
| 00367 | 00856 | E |
| 00367 | 00857 | E |
| 00367 | 00858 | E |
| 00367 | 00860 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00367 | 00861 | E |
| 00367 | 00862 | E |
| 00367 | 00863 | E |
| 00367 | 00864 | E |
| 00367 | 00865 | E |
| 00367 | 00866 | E |
| 00367 | 00868 | E |
| 00367 | 00869 | E |
| 00367 | 00870 | E |
| 00367 | 00871 | E |
| 00367 | 00874 | E |
| 00367 | 00875 | E |
| 00367 | 00880 | E |
| 00367 | 00891 | E |
| 00367 | 00895 | E |
| 00367 | 00896 | E |
| 00367 | 00897 | E |
| 00367 | 00903 | E |
| 00367 | 00904 | E |
| 00367 | 00905 | E |
| 00367 | 00912 | E |
| 00367 | 00915 | E |
| 00367 | 00916 | E |
| 00367 | 00918 | E |
| 00367 | 00920 | E |
| 00367 | 00921 | E |
| 00367 | 00922 | E |
| 00367 | 00923 | E |
| 00367 | 00924 | E |
| 00367 | 01004 | E |
| 00367 | 01006 | E |
| 00367 | 01008 | E |
| 00367 | 01009 | R |
| 00367 | 01010 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00367 | 01011 | E |
| 00367 | 01012 | E |
| 00367 | 01013 | E |
| 00367 | 01014 | E |
| 00367 | 01015 | E |
| 00367 | 01016 | E |
| 00367 | 01017 | E |
| 00367 | 01018 | E |
| 00367 | 01019 | E |
| 00367 | 01020 | E |
| 00367 | 01021 | E |
| 00367 | 01023 | E |
| 00367 | 01025 | E |
| 00367 | 01026 | E |
| 00367 | 01027 | E |
| 00367 | 01040 | E |
| 00367 | 01041 | E |
| 00367 | 01042 | E |
| 00367 | 01043 | E |
| 00367 | 01046 | E |
| 00367 | 01047 | E |
| 00367 | 01051 | E |
| 00367 | 01088 | E |
| 00367 | 01089 | E |
| 00367 | 01097 | E |
| 00367 | 01098 | E |
| 00367 | 01100 | E |
| 00367 | 01101 | E |
| 00367 | 01102 | E |
| 00367 | 01103 | E |
| 00367 | 01104 | E |
| 00367 | 01105 | E |
| 00367 | 01106 | E |
| 00367 | 01107 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00367 | 01112 | E |
| 00367 | 01114 | E |
| 00367 | 01115 | E |
| 00367 | 01122 | E |
| 00367 | 01123 | E |
| 00367 | 01124 | E |
| 00367 | 01125 | E |
| 00367 | 01126 | E |
| 00367 | 01131 | E |
| 00367 | 01140 | E |
| 00367 | 01141 | E |
| 00367 | 01142 | E |
| 00367 | 01143 | E |
| 00367 | 01144 | E |
| 00367 | 01145 | E |
| 00367 | 01146 | E |
| 00367 | 01147 | E |
| 00367 | 01148 | E |
| 00367 | 01149 | E |
| 00367 | 01250 | E |
| 00367 | 01251 | E |
| 00367 | 01252 | E |
| 00367 | 01253 | E |
| 00367 | 01254 | E |
| 00367 | 01255 | E |
| 00367 | 01256 | E |
| 00367 | 01257 | E |
| 00367 | 01275 | E |
| 00367 | 01276 | E |
| 00367 | 01277 | E |
| 00367 | 01280 | E |
| 00367 | 01281 | E |
| 00367 | 01282 | E |
| 00367 | 01283 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00367 | 04133 | E |
| 00367 | 04369 | E |
| 00367 | 04371 | E |
| 00367 | 04373 | E |
| 00367 | 04374 | E |
| 00367 | 04376 | E |
| 00367 | 04378 | E |
| 00367 | 04380 | E |
| 00367 | 04381 | E |
| 00367 | 04386 | E |
| 00367 | 04516 | E |
| 00367 | 04519 | E |
| 00367 | 04520 | E |
| 00367 | 04533 | E |
| 00367 | 04596 | E |
| 00367 | 04929 | E |
| 00367 | 04932 | E |
| 00367 | 04934 | E |
| 00367 | 04946 | E |
| 00367 | 04947 | E |
| 00367 | 04949 | E |
| 00367 | 04953 | E |
| 00367 | 04964 | E |
| 00367 | 04965 | E |
| 00367 | 04966 | E |
| 00367 | 04967 | E |
| 00367 | 04970 | E |
| 00367 | 04976 | E |
| 00367 | 04992 | E |
| 00367 | 04993 | E |
| 00367 | 05014 | E |
| 00367 | 05100 | E |
| 00367 | 05137 | E |
| 00367 | 05143 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00367 | 05211 | E |
| 00367 | 08229 | E |
| 00367 | 08448 | E |
| 00367 | 08629 | E |
| 00367 | 08692 | E |
| 00367 | 09025 | E |
| 00367 | 09028 | E |
| 00367 | 09047 | E |
| 00367 | 09060 | E |
| 00367 | 09089 | E |
| 00367 | 12544 | E |
| 00367 | 12725 | E |
| 00367 | 12788 | E |
| 00367 | 13152 | E |
| 00367 | 16421 | E |
| 00367 | 16821 | E |
| 00367 | 16884 | E |
| 00367 | 20517 | E |
| 00367 | 20917 | E |
| 00367 | 20980 | E |
| 00367 | 24613 | E |
| 00367 | 25013 | E |
| 00367 | 25076 | E |
| 00367 | 25426 | E |
| 00367 | 25427 | E |
| 00367 | 25428 | E |
| 00367 | 25429 | E |
| 00367 | 25431 | E |
| 00367 | 25432 | E |
| 00367 | 25433 | E |
| 00367 | 25436 | E |
| 00367 | 25437 | E |
| 00367 | 25438 | E |
| 00367 | 25439 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00367 | 25440 | E |
| 00367 | 25441 | E |
| 00367 | 25442 | E |
| 00367 | 25444 | E |
| 00367 | 25445 | E |
| 00367 | 25450 | E |
| 00367 | 25467 | E |
| 00367 | 25473 | E |
| 00367 | 25479 | E |
| 00367 | 25480 | E |
| 00367 | 25580 | E |
| 00367 | 25616 | E |
| 00367 | 25617 | E |
| 00367 | 25618 | E |
| 00367 | 25619 | E |
| 00367 | 25664 | E |
| 00367 | 25690 | E |
| 00367 | 25691 | E |
| 00367 | 29109 | E |
| 00367 | 29172 | E |
| 00367 | 29522 | E |
| 00367 | 29523 | E |
| 00367 | 29524 | E |
| 00367 | 29525 | E |
| 00367 | 29527 | E |
| 00367 | 29528 | E |
| 00367 | 29529 | E |
| 00367 | 29532 | E |
| 00367 | 29533 | E |
| 00367 | 29534 | E |
| 00367 | 29535 | E |
| 00367 | 29536 | E |
| 00367 | 29537 | E |
| 00367 | 29540 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00367 | 29541 | E |
| 00367 | 29546 | E |
| 00367 | 29712 | E |
| 00367 | 29713 | E |
| 00367 | 29714 | E |
| 00367 | 29715 | E |
| 00367 | 29760 | E |
| 00367 | 32805 | E |
| 00367 | 33058 | E |
| 00367 | 33205 | E |
| 00367 | 33268 | E |
| 00367 | 33618 | E |
| 00367 | 33619 | E |
| 00367 | 33620 | E |
| 00367 | 33621 | E |
| 00367 | 33623 | E |
| 00367 | 33624 | E |
| 00367 | 33632 | E |
| 00367 | 33636 | E |
| 00367 | 33637 | E |
| 00367 | 33665 | E |
| 00367 | 37301 | E |
| 00367 | 37719 | E |
| 00367 | 37728 | E |
| 00367 | 37732 | E |
| 00367 | 37761 | E |
| 00367 | 41397 | E |
| 00367 | 41460 | E |
| 00367 | 41824 | E |
| 00367 | 41828 | E |
| 00367 | 45493 | E |
| 00367 | 45556 | E |
| 00367 | 45920 | E |
| 00367 | 49589 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00367 | 49652 | E |
| 00367 | 53748 | E |
| 00367 | 61696 | E |
| 00367 | 61697 | E |
| 00367 | 61698 | E |
| 00367 | 61699 | E |
| 00367 | 61710 | E |
| 00367 | 61711 | E |
| 00367 | 61712 | E |
| 00420 | 00037 | R,E |
| 00420 | 00256 | R |
| 00420 | 00367 | E |
| 00420 | 00424 | R |
| 00420 | 00425 | C |
| 00420 | 00437 | R,E,M |
| 00420 | 00500 | R,E |
| 00420 | 00720 | C |
| 00420 | 00737 | R |
| 00420 | 00775 | R |
| 00420 | 00819 | R,L |
| 00420 | 00850 | R,M |
| 00420 | 00852 | R,E,M |
| 00420 | 00857 | R,E |
| 00420 | 00860 | R,E |
| 00420 | 00861 | R,E,M |
| 00420 | 00862 | R,E,M |
| 00420 | 00863 | R,E |
| 00420 | 00864 | R,E,M |
| 00420 | 00865 | R,E |
| 00420 | 01008 | R |
| 00420 | 01046 | C,L |
| 00420 | 01051 | E |
| 00420 | 01089 | C,L |
| 00420 | 01098 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00420 | 01112 | R |
| 00420 | 01122 | R |
| 00420 | 01127 | R |
| 00420 | 01252 | R,L |
| 00420 | 01256 | C,L |
| 00420 | 05352 | C,L |
| 00420 | 09238 | E,L |
| 00420 | 17248 | R,E,L |
| 00421 | 00367 | E |
| 00423 | 00037 | R,E |
| 00423 | 00256 | R |
| 00423 | 00273 | R |
| 00423 | 00277 | R |
| 00423 | 00278 | R |
| 00423 | 00280 | R |
| 00423 | 00284 | R |
| 00423 | 00285 | R |
| 00423 | 00297 | R |
| 00423 | 00367 | E |
| 00423 | 00437 | R,E |
| 00423 | 00500 | R,E |
| 00423 | 00737 | R,E |
| 00423 | 00775 | R,E |
| 00423 | 00813 | R |
| 00423 | 00819 | R |
| 00423 | 00838 | R |
| 00423 | 00850 | R |
| 00423 | 00851 | R |
| 00423 | 00852 | R,E |
| 00423 | 00857 | R,E |
| 00423 | 00860 | R,E |
| 00423 | 00861 | R,E |
| 00423 | 00862 | R,E |
| 00423 | 00863 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00423 | 00864 | R,E |
| 00423 | 00865 | R,E |
| 00423 | 00869 | R |
| 00423 | 00870 | R |
| 00423 | 00871 | R |
| 00423 | 00874 | R |
| 00423 | 00875 | R |
| 00423 | 00880 | R |
| 00423 | 00897 | R |
| 00423 | 00903 | R |
| 00423 | 00912 | R |
| 00423 | 00916 | R |
| 00423 | 00920 | R |
| 00423 | 01009 | E |
| 00423 | 01025 | R |
| 00423 | 01026 | R |
| 00423 | 01027 | R |
| 00423 | 01041 | R |
| 00423 | 01042 | R |
| 00423 | 01043 | R |
| 00423 | 01051 | E |
| 00423 | 01112 | R |
| 00423 | 01122 | R |
| 00423 | 01252 | R |
| 00423 | 01253 | R,E |
| 00423 | 01280 | R |
| 00423 | 09061 | R,E |
| 00424 | 00037 | R,E |
| 00424 | 00256 | R |
| 00424 | 00367 | E |
| 00424 | 00420 | R |
| 00424 | 00437 | R,E,M |
| 00424 | 00500 | R,E |
| 00424 | 00737 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00424 | 00775 | R |
| 00424 | 00803 | R |
| 00424 | 00819 | R,L |
| 00424 | 00836 | E |
| 00424 | 00850 | R,E,M |
| 00424 | 00852 | R,E,M |
| 00424 | 00856 | R,M |
| 00424 | 00857 | R,E |
| 00424 | 00860 | R,E |
| 00424 | 00861 | R,E,M |
| 00424 | 00862 | R,E,M |
| 00424 | 00863 | R,E |
| 00424 | 00864 | R,E,M |
| 00424 | 00865 | R,E |
| 00424 | 00867 | R,M |
| 00424 | 00916 | R,E,L |
| 00424 | 01051 | E |
| 00424 | 01112 | R |
| 00424 | 01122 | R |
| 00424 | 01252 | R,L |
| 00424 | 01255 | R,E,L |
| 00424 | 05351 | R,E,L |
| 00424 | 09048 | R |
| 00425 | 00037 | R,E |
| 00425 | 00420 | C |
| 00425 | 00500 | R,E |
| 00425 | 00720 | E |
| 00425 | 00819 | R,E,L |
| 00425 | 00864 | C,L |
| 00425 | 01046 | C,L |
| 00425 | 01089 | E,L |
| 00425 | 01140 | R,E |
| 00425 | 01148 | R,E |
| 00425 | 01256 | E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00425 | 05348 | R,E,L |
| 00425 | 05352 | R,E,L |
| 00425 | 16804 | C |
| 00437 | 00037 | R,E,M |
| 00437 | 00256 | R,E |
| 00437 | 00259 | E |
| 00437 | 00273 | R,E,M |
| 00437 | 00275 | R,E,M |
| 00437 | 00277 | R,E,M |
| 00437 | 00278 | R,E,M |
| 00437 | 00280 | R,E,M |
| 00437 | 00284 | R,E,M |
| 00437 | 00285 | R,E,M |
| 00437 | 00290 | R,E,M |
| 00437 | 00297 | R,E,M |
| 00437 | 00367 | E |
| 00437 | 00420 | R,E,M |
| 00437 | 00423 | R,E |
| 00437 | 00424 | R,E,M |
| 00437 | 00500 | R,E,M |
| 00437 | 00737 | R |
| 00437 | 00775 | R,E |
| 00437 | 00813 | R |
| 00437 | 00819 | R |
| 00437 | 00833 | R,E,M |
| 00437 | 00836 | E,M |
| 00437 | 00838 | R,E,M |
| 00437 | 00850 | R,E |
| 00437 | 00852 | R |
| 00437 | 00855 | R |
| 00437 | 00857 | R |
| 00437 | 00858 | R,E |
| 00437 | 00860 | R |
| 00437 | 00861 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00437 | 00862 | R |
| 00437 | 00863 | R |
| 00437 | 00865 | R |
| 00437 | 00866 | R |
| 00437 | 00869 | R |
| 00437 | 00870 | R,E,M |
| 00437 | 00871 | R,E,M |
| 00437 | 00874 | R |
| 00437 | 00875 | R,E,M |
| 00437 | 00880 | R,E,M |
| 00437 | 00897 | R,E |
| 00437 | 00903 | R |
| 00437 | 00905 | R,E |
| 00437 | 00912 | R |
| 00437 | 00914 | R |
| 00437 | 00915 | R |
| 00437 | 00916 | R |
| 00437 | 00920 | R |
| 00437 | 00921 | R |
| 00437 | 00922 | R |
| 00437 | 00923 | R,E |
| 00437 | 00924 | R,E,M |
| 00437 | 01025 | R,E,M |
| 00437 | 01026 | R,E,M |
| 00437 | 01027 | R,E,M |
| 00437 | 01040 | R,E |
| 00437 | 01041 | R,E |
| 00437 | 01042 | R |
| 00437 | 01043 | R,E |
| 00437 | 01047 | R,E,M |
| 00437 | 01051 | R |
| 00437 | 01097 | R,E |
| 00437 | 01098 | R |
| 00437 | 01114 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00437 | 01115 | E |
| 00437 | 01126 | E |
| 00437 | 01140 | R,E,M |
| 00437 | 01141 | R,E,M |
| 00437 | 01142 | R,E,M |
| 00437 | 01143 | R,E,M |
| 00437 | 01144 | R,E,M |
| 00437 | 01145 | R,E,M |
| 00437 | 01146 | R,E,M |
| 00437 | 01147 | R,E,M |
| 00437 | 01148 | R,E,M |
| 00437 | 01149 | R,E,M |
| 00437 | 01252 | R |
| 00437 | 01257 | R |
| 00437 | 01275 | R |
| 00437 | 01280 | R |
| 00437 | 01281 | R |
| 00437 | 01283 | R |
| 00437 | 04946 | E |
| 00437 | 05348 | R,E |
| 00437 | 28709 | R,E,M |
| 00500 | 00037 | R,E |
| 00500 | 00256 | R,E |
| 00500 | 00273 | R,E |
| 00500 | 00274 | R |
| 00500 | 00275 | R |
| 00500 | 00277 | R,E |
| 00500 | 00278 | R,E |
| 00500 | 00280 | R,E |
| 00500 | 00281 | R,E |
| 00500 | 00282 | R |
| 00500 | 00284 | R,E |
| 00500 | 00285 | R,E |
| 00500 | 00290 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00500 | 00297 | R,E |
| 00500 | 00367 | E |
| 00500 | 00420 | R,E |
| 00500 | 00423 | R,E |
| 00500 | 00424 | R,E |
| 00500 | 00425 | R,E |
| 00500 | 00437 | R,E,M,P |
| 00500 | 00737 | R,E |
| 00500 | 00775 | R,E |
| 00500 | 00813 | R,E,L |
| 00500 | 00819 | R,L |
| 00500 | 00833 | R,E |
| 00500 | 00836 | R,E |
| 00500 | 00838 | E |
| 00500 | 00850 | R,E,M |
| 00500 | 00851 | R |
| 00500 | 00852 | R,E,M |
| 00500 | 00855 | R,M |
| 00500 | 00856 | R,M |
| 00500 | 00857 | R,E |
| 00500 | 00858 | R,E,M |
| 00500 | 00860 | R,E |
| 00500 | 00861 | R,E,M |
| 00500 | 00862 | R,E,M |
| 00500 | 00863 | R,E |
| 00500 | 00864 | R,E,M |
| 00500 | 00865 | R,E |
| 00500 | 00866 | E,L |
| 00500 | 00869 | R,E,M |
| 00500 | 00870 | R,E |
| 00500 | 00871 | R,E |
| 00500 | 00875 | R,E |
| 00500 | 00880 | R,E |
| 00500 | 00891 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00500 | 00895 | E |
| 00500 | 00897 | E |
| 00500 | 00901 | R,E,L |
| 00500 | 00902 | R,E,L |
| 00500 | 00903 | E |
| 00500 | 00904 | E,L |
| 00500 | 00905 | R,E |
| 00500 | 00912 | R,E,L |
| 00500 | 00914 | R,L |
| 00500 | 00915 | R,L |
| 00500 | 00916 | R,E,L |
| 00500 | 00920 | R,E,L |
| 00500 | 00921 | R,L |
| 00500 | 00922 | R,L |
| 00500 | 00923 | R,E,L |
| 00500 | 00924 | R,E |
| 00500 | 01004 | R |
| 00500 | 01009 | E |
| 00500 | 01010 | E |
| 00500 | 01011 | E |
| 00500 | 01012 | E |
| 00500 | 01013 | E |
| 00500 | 01014 | E |
| 00500 | 01015 | E |
| 00500 | 01016 | E |
| 00500 | 01017 | E |
| 00500 | 01018 | E |
| 00500 | 01019 | E |
| 00500 | 01020 | E |
| 00500 | 01021 | E |
| 00500 | 01023 | E |
| 00500 | 01025 | R,E |
| 00500 | 01026 | R,E |
| 00500 | 01027 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00500 | 01040 | R,E |
| 00500 | 01041 | R,E |
| 00500 | 01042 | R,E |
| 00500 | 01043 | R,E |
| 00500 | 01046 | E,L |
| 00500 | 01047 | R |
| 00500 | 01051 | R,E |
| 00500 | 01088 | R,E,L |
| 00500 | 01089 | R,E,L |
| 00500 | 01097 | R,E |
| 00500 | 01100 | R,E |
| 00500 | 01101 | E |
| 00500 | 01102 | E |
| 00500 | 01103 | E |
| 00500 | 01104 | E |
| 00500 | 01105 | E |
| 00500 | 01106 | E |
| 00500 | 01107 | E |
| 00500 | 01112 | R,E |
| 00500 | 01114 | E |
| 00500 | 01115 | E,L |
| 00500 | 01122 | R |
| 00500 | 01123 | R,E |
| 00500 | 01124 | R,E,L |
| 00500 | 01125 | R,E,L |
| 00500 | 01126 | E,L |
| 00500 | 01129 | R,E |
| 00500 | 01130 | R,E |
| 00500 | 01131 | R |
| 00500 | 01132 | R,E |
| 00500 | 01133 | R,E |
| 00500 | 01137 | E |
| 00500 | 01140 | R,E |
| 00500 | 01141 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00500 | 01142 | R,E |
| 00500 | 01143 | R,E |
| 00500 | 01144 | R,E |
| 00500 | 01145 | R,E |
| 00500 | 01146 | R,E |
| 00500 | 01147 | R,E |
| 00500 | 01148 | E |
| 00500 | 01149 | R,E |
| 00500 | 01250 | R,E,L |
| 00500 | 01251 | R,E,L |
| 00500 | 01252 | R,E,L |
| 00500 | 01253 | R,E,L |
| 00500 | 01254 | R,E,L |
| 00500 | 01255 | R,E,L |
| 00500 | 01256 | R,E,L |
| 00500 | 01257 | R |
| 00500 | 01258 | R,E |
| 00500 | 01275 | R |
| 00500 | 01280 | R |
| 00500 | 01281 | R |
| 00500 | 01282 | R |
| 00500 | 01283 | R |
| 00500 | 04909 | R,E,L |
| 00500 | 05348 | R,E,L |
| 00500 | 05350 | R,L |
| 00500 | 09049 | E |
| 00720 | 00037 | R |
| 00720 | 00420 | C |
| 00720 | 00425 | E |
| 00720 | 00864 | C |
| 00720 | 01046 | C |
| 00720 | 01256 | C |
| 00737 | 00037 | R |
| 00737 | 00256 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00737 | 00273 | R |
| 00737 | 00277 | R |
| 00737 | 00278 | R |
| 00737 | 00280 | R |
| 00737 | 00284 | R |
| 00737 | 00285 | R |
| 00737 | 00290 | E |
| 00737 | 00297 | R |
| 00737 | 00420 | R |
| 00737 | 00423 | R,E |
| 00737 | 00424 | R |
| 00737 | 00437 | R |
| 00737 | 00500 | R,E |
| 00737 | 00813 | R,E |
| 00737 | 00833 | E |
| 00737 | 00836 | E |
| 00737 | 00838 | E |
| 00737 | 00850 | R |
| 00737 | 00869 | R,E |
| 00737 | 00870 | R |
| 00737 | 00871 | R |
| 00737 | 00875 | R,E |
| 00737 | 00880 | R |
| 00737 | 00905 | R |
| 00737 | 01025 | R |
| 00737 | 01026 | R |
| 00737 | 01027 | E |
| 00737 | 01097 | R |
| 00737 | 01252 | R |
| 00737 | 01253 | R,E |
| 00737 | 01280 | R,E |
| 00737 | 01287 | R,E |
| 00737 | 28709 | E |
| 00775 | 00037 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00775 | 00256 | R,E |
| 00775 | 00273 | R |
| 00775 | 00277 | R,E |
| 00775 | 00278 | R |
| 00775 | 00280 | R,E |
| 00775 | 00284 | R |
| 00775 | 00285 | R,E |
| 00775 | 00290 | E |
| 00775 | 00297 | R,E |
| 00775 | 00420 | R |
| 00775 | 00423 | R,E |
| 00775 | 00424 | R |
| 00775 | 00437 | R,E |
| 00775 | 00500 | R,E |
| 00775 | 00833 | E |
| 00775 | 00836 | E |
| 00775 | 00838 | E |
| 00775 | 00850 | R |
| 00775 | 00870 | R,E |
| 00775 | 00871 | R |
| 00775 | 00875 | R,E |
| 00775 | 00880 | R |
| 00775 | 00905 | R,E |
| 00775 | 01025 | R |
| 00775 | 01026 | R,E |
| 00775 | 01027 | E |
| 00775 | 01097 | R,E |
| 00775 | 01112 | R |
| 00775 | 01122 | R |
| 00775 | 01252 | R,E |
| 00775 | 01257 | R |
| 00775 | 28709 | E |
| 00803 | 00367 | E |
| 00803 | 00424 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00803 | 00819 | R,E |
| 00803 | 00850 | R,E |
| 00803 | 00856 | R |
| 00803 | 00862 | R,E |
| 00803 | 00916 | R,E,L |
| 00803 | 01252 | R,E |
| 00803 | 01255 | R,E |
| 00806 | 01137 | E |
| 00808 | 00259 | E |
| 00808 | 00858 | R,E |
| 00808 | 00859 | R,E |
| 00808 | 00872 | R,E |
| 00808 | 00923 | R,E |
| 00808 | 00924 | R,E,M |
| 00808 | 01025 | R,E,M |
| 00808 | 01140 | R,E,M |
| 00808 | 01148 | R,E,M |
| 00808 | 01153 | R,E,M |
| 00808 | 01154 | R,E,M |
| 00808 | 01158 | R,M |
| 00808 | 05347 | R,E |
| 00808 | 05348 | R,E |
| 00813 | 00037 | R,L |
| 00813 | 00273 | R,L |
| 00813 | 00277 | R,L |
| 00813 | 00278 | R,L |
| 00813 | 00280 | R,L |
| 00813 | 00284 | R,L |
| 00813 | 00285 | R,L |
| 00813 | 00297 | R,L |
| 00813 | 00367 | E |
| 00813 | 00423 | R |
| 00813 | 00437 | R |
| 00813 | 00500 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00813 | 00737 | R,E |
| 00813 | 00819 | R |
| 00813 | 00838 | R,L |
| 00813 | 00850 | R |
| 00813 | 00852 | R |
| 00813 | 00857 | R |
| 00813 | 00860 | R |
| 00813 | 00861 | R |
| 00813 | 00863 | R |
| 00813 | 00869 | R,E |
| 00813 | 00870 | R,L |
| 00813 | 00871 | R,L |
| 00813 | 00874 | R |
| 00813 | 00875 | R,L |
| 00813 | 00880 | R,L |
| 00813 | 00897 | R |
| 00813 | 00903 | R |
| 00813 | 00912 | R |
| 00813 | 00916 | R |
| 00813 | 00920 | R |
| 00813 | 01025 | R,L |
| 00813 | 01026 | R,L |
| 00813 | 01027 | R,L |
| 00813 | 01041 | R |
| 00813 | 01042 | R |
| 00813 | 01043 | R |
| 00813 | 01252 | R |
| 00813 | 01253 | R |
| 00813 | 01280 | R |
| 00813 | 01287 | R,E |
| 00813 | 05349 | R,E |
| 00819 | 00037 | R,L |
| 00819 | 00256 | R |
| 00819 | 00273 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00819 | 00274 | R,E,L |
| 00819 | 00275 | R,E,L |
| 00819 | 00277 | R,L |
| 00819 | 00278 | R,L |
| 00819 | 00280 | R,L |
| 00819 | 00281 | R,E,L |
| 00819 | 00282 | R,E,L |
| 00819 | 00284 | R,L |
| 00819 | 00285 | R,L |
| 00819 | 00290 | E |
| 00819 | 00297 | R,L |
| 00819 | 00367 | E |
| 00819 | 00420 | R,L |
| 00819 | 00423 | R |
| 00819 | 00424 | R,L |
| 00819 | 00425 | R,E,L |
| 00819 | 00437 | R |
| 00819 | 00500 | R,L |
| 00819 | 00803 | R,E |
| 00819 | 00813 | R |
| 00819 | 00833 | E,L |
| 00819 | 00836 | E,L |
| 00819 | 00838 | R,L |
| 00819 | 00850 | R,E |
| 00819 | 00852 | R |
| 00819 | 00855 | R |
| 00819 | 00857 | R |
| 00819 | 00858 | R,E |
| 00819 | 00860 | R |
| 00819 | 00861 | R |
| 00819 | 00863 | R |
| 00819 | 00864 | R |
| 00819 | 00865 | R |
| 00819 | 00866 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00819 | 00869 | R |
| 00819 | 00870 | R,L |
| 00819 | 00871 | R,L |
| 00819 | 00874 | R |
| 00819 | 00875 | R,L |
| 00819 | 00880 | R,L |
| 00819 | 00897 | R |
| 00819 | 00903 | R |
| 00819 | 00905 | R |
| 00819 | 00912 | R |
| 00819 | 00914 | R |
| 00819 | 00915 | R |
| 00819 | 00916 | R |
| 00819 | 00920 | R |
| 00819 | 00921 | R |
| 00819 | 00922 | R |
| 00819 | 00923 | E |
| 00819 | 00924 | R,E,L |
| 00819 | 01004 | R |
| 00819 | 01025 | R,L |
| 00819 | 01026 | R,L |
| 00819 | 01027 | R,E,L |
| 00819 | 01041 | R,E |
| 00819 | 01042 | R |
| 00819 | 01043 | R |
| 00819 | 01047 | R,L |
| 00819 | 01051 | R |
| 00819 | 01088 | R |
| 00819 | 01089 | R |
| 00819 | 01097 | R |
| 00819 | 01098 | R |
| 00819 | 01112 | R,E,L |
| 00819 | 01114 | R,E |
| 00819 | 01122 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00819 | 01123 | R,E,L |
| 00819 | 01126 | E |
| 00819 | 01130 | R,E |
| 00819 | 01132 | R,E |
| 00819 | 01137 | E |
| 00819 | 01140 | R,E,L |
| 00819 | 01141 | R,E,L |
| 00819 | 01142 | R,E,L |
| 00819 | 01143 | R,E,L |
| 00819 | 01144 | R,E,L |
| 00819 | 01145 | R,E,L |
| 00819 | 01146 | R,E,L |
| 00819 | 01147 | R,E,L |
| 00819 | 01148 | R,E,L |
| 00819 | 01149 | R,E,L |
| 00819 | 01153 | R,E,L |
| 00819 | 01154 | R,E,L |
| 00819 | 01155 | R,E,L |
| 00819 | 01156 | R,E,L |
| 00819 | 01157 | R,E,L |
| 00819 | 01158 | R,E,L |
| 00819 | 01160 | R,E,L |
| 00819 | 01164 | R,E |
| 00819 | 01250 | R |
| 00819 | 01251 | R |
| 00819 | 01252 | R |
| 00819 | 01253 | R |
| 00819 | 01254 | R |
| 00819 | 01255 | R |
| 00819 | 01257 | R |
| 00819 | 01258 | R |
| 00819 | 01275 | R |
| 00819 | 01280 | R |
| 00819 | 01281 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00819 | 01283 | R |
| 00819 | 05348 | R,E |
| 00833 | 00037 | R,E |
| 00833 | 00256 | E |
| 00833 | 00273 | R,E |
| 00833 | 00277 | R,E |
| 00833 | 00278 | R,E |
| 00833 | 00280 | R,E |
| 00833 | 00284 | R,E |
| 00833 | 00285 | R,E |
| 00833 | 00290 | R,E |
| 00833 | 00297 | R,E |
| 00833 | 00367 | E |
| 00833 | 00437 | R,E,M |
| 00833 | 00500 | R,E |
| 00833 | 00737 | E |
| 00833 | 00775 | E |
| 00833 | 00819 | E,L |
| 00833 | 00836 | R,E |
| 00833 | 00850 | R,E,M |
| 00833 | 00852 | R,E,M |
| 00833 | 00855 | R,E,M |
| 00833 | 00857 | R,E |
| 00833 | 00860 | R,E |
| 00833 | 00861 | R,E,M |
| 00833 | 00862 | R,E,M |
| 00833 | 00863 | R,E |
| 00833 | 00864 | R,E,M |
| 00833 | 00865 | R,E |
| 00833 | 00870 | R,E |
| 00833 | 00871 | R,E |
| 00833 | 00891 | E |
| 00833 | 01009 | E |
| 00833 | 01025 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00833 | 01026 | R,E |
| 00833 | 01027 | R,E |
| 00833 | 01040 | R,E |
| 00833 | 01041 | R,E |
| 00833 | 01042 | R |
| 00833 | 01043 | R,E |
| 00833 | 01047 | R,E |
| 00833 | 01088 | R,E,L |
| 00833 | 01112 | R |
| 00833 | 01122 | R |
| 00833 | 01126 | E,L |
| 00833 | 01252 | E,L |
| 00834 | 00926 | E |
| 00834 | 00951 | E |
| 00834 | 00971 | E |
| 00834 | 01362 | E |
| 00834 | 04930 | E |
| 00835 | 00927 | E |
| 00835 | 00947 | E |
| 00836 | 00037 | R,E |
| 00836 | 00256 | E |
| 00836 | 00273 | R,E |
| 00836 | 00277 | R,E |
| 00836 | 00278 | R,E |
| 00836 | 00280 | R,E |
| 00836 | 00284 | R,E |
| 00836 | 00285 | R,E |
| 00836 | 00290 | R,E |
| 00836 | 00297 | R,E |
| 00836 | 00367 | E |
| 00836 | 00424 | E |
| 00836 | 00437 | E,M |
| 00836 | 00500 | R,E |
| 00836 | 00737 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00836 | 00775 | E |
| 00836 | 00819 | E,L |
| 00836 | 00833 | R,E |
| 00836 | 00850 | R,E,M |
| 00836 | 00852 | R,M |
| 00836 | 00855 | R,M |
| 00836 | 00857 | R |
| 00836 | 00870 | R |
| 00836 | 00871 | R,E |
| 00836 | 00875 | R,E |
| 00836 | 00903 | E |
| 00836 | 01009 | E |
| 00836 | 01025 | R |
| 00836 | 01026 | R |
| 00836 | 01027 | R,E |
| 00836 | 01040 | R |
| 00836 | 01041 | R |
| 00836 | 01042 | R,E |
| 00836 | 01043 | R |
| 00836 | 01047 | R,E |
| 00836 | 01088 | R,L |
| 00836 | 01112 | R |
| 00836 | 01114 | E |
| 00836 | 01115 | E,L |
| 00836 | 01122 | R |
| 00836 | 01252 | E,L |
| 00837 | 00928 | E |
| 00837 | 01380 | E |
| 00837 | 01382 | E |
| 00837 | 01385 | E |
| 00837 | 04933 | E |
| 00837 | 13125 | E |
| 00838 | 00037 | E |
| 00838 | 00256 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00838 | 00273 | E |
| 00838 | 00277 | E |
| 00838 | 00278 | E |
| 00838 | 00280 | E |
| 00838 | 00284 | E |
| 00838 | 00285 | E |
| 00838 | 00297 | E |
| 00838 | 00367 | E |
| 00838 | 00423 | R |
| 00838 | 00437 | R,E,M |
| 00838 | 00500 | E |
| 00838 | 00737 | E |
| 00838 | 00775 | E |
| 00838 | 00813 | R,L |
| 00838 | 00819 | R,L |
| 00838 | 00850 | R,E,M |
| 00838 | 00852 | R,E,M |
| 00838 | 00857 | R,E |
| 00838 | 00860 | R,E |
| 00838 | 00861 | R,E,M |
| 00838 | 00862 | R,E,M |
| 00838 | 00863 | R,E |
| 00838 | 00864 | R,E,M |
| 00838 | 00865 | R,E |
| 00838 | 00869 | R,M |
| 00838 | 00870 | R |
| 00838 | 00871 | E |
| 00838 | 00874 | R,E,M |
| 00838 | 00875 | R |
| 00838 | 00880 | R |
| 00838 | 00897 | R |
| 00838 | 00903 | R |
| 00838 | 00912 | R,L |
| 00838 | 00916 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00838 | 00920 | R,L |
| 00838 | 01025 | R |
| 00838 | 01026 | R |
| 00838 | 01027 | R |
| 00838 | 01041 | R |
| 00838 | 01042 | R |
| 00838 | 01043 | R |
| 00838 | 01051 | E |
| 00838 | 01112 | R |
| 00838 | 01122 | R |
| 00838 | 01161 | R,E,L |
| 00838 | 01252 | E,L |
| 00848 | 00924 | R,E,M |
| 00848 | 01123 | R,E,M |
| 00848 | 01148 | R,E,M |
| 00848 | 01154 | R,M |
| 00848 | 01158 | R,E,M |
| 00848 | 05347 | R,E |
| 00849 | 00924 | R,E |
| 00849 | 01025 | R,E |
| 00849 | 01148 | R,E |
| 00849 | 01154 | R,E |
| 00849 | 01158 | R |
| 00849 | 05347 | R,E |
| 00850 | 00037 | R,E,C,M |
| 00850 | 00256 | R,E |
| 00850 | 00259 | E |
| 00850 | 00273 | R,E,C,M |
| 00850 | 00274 | R,E,M |
| 00850 | 00275 | R,E,M |
| 00850 | 00277 | R,E,C,M |
| 00850 | 00278 | R,E,C,M |
| 00850 | 00280 | R,E,C,M |
| 00850 | 00284 | R,E,C,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00850 | 00285 | R,E,C,M |
| 00850 | 00290 | R,E,M |
| 00850 | 00297 | R,E,C,M |
| 00850 | 00367 | E |
| 00850 | 00420 | R,M |
| 00850 | 00423 | R |
| 00850 | 00424 | R,E,M |
| 00850 | 00437 | R,E |
| 00850 | 00500 | R,E,C,M |
| 00850 | 00737 | R |
| 00850 | 00775 | R |
| 00850 | 00803 | R,E |
| 00850 | 00813 | R |
| 00850 | 00819 | R,E |
| 00850 | 00833 | R,E,M |
| 00850 | 00836 | R,E,M |
| 00850 | 00838 | R,E,M |
| 00850 | 00852 | R |
| 00850 | 00855 | R |
| 00850 | 00856 | R |
| 00850 | 00857 | R |
| 00850 | 00858 | E |
| 00850 | 00860 | R |
| 00850 | 00861 | R |
| 00850 | 00862 | R |
| 00850 | 00863 | R |
| 00850 | 00864 | R |
| 00850 | 00865 | R |
| 00850 | 00866 | R |
| 00850 | 00869 | R |
| 00850 | 00870 | R,E,M |
| 00850 | 00871 | R,E,C,M |
| 00850 | 00874 | R |
| 00850 | 00875 | R,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00850 | 00880 | R,E,M |
| 00850 | 00897 | R,E |
| 00850 | 00903 | R |
| 00850 | 00905 | R,E |
| 00850 | 00912 | R |
| 00850 | 00914 | R |
| 00850 | 00915 | R |
| 00850 | 00916 | R |
| 00850 | 00920 | R |
| 00850 | 00921 | R |
| 00850 | 00922 | R |
| 00850 | 00923 | R,E |
| 00850 | 00924 | R,E,M |
| 00850 | 01004 | R |
| 00850 | 01025 | R,M |
| 00850 | 01026 | R,E,M |
| 00850 | 01027 | R,E,M |
| 00850 | 01040 | R,E |
| 00850 | 01041 | R,E |
| 00850 | 01042 | R |
| 00850 | 01043 | R,E |
| 00850 | 01047 | R,C,M |
| 00850 | 01051 | R |
| 00850 | 01088 | R |
| 00850 | 01089 | R |
| 00850 | 01097 | R |
| 00850 | 01098 | R |
| 00850 | 01100 | R |
| 00850 | 01112 | R,M |
| 00850 | 01114 | R,E |
| 00850 | 01122 | R,M |
| 00850 | 01126 | E |
| 00850 | 01130 | R,E |
| 00850 | 01132 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00850 | 01140 | R,E,M |
| 00850 | 01141 | R,E,M |
| 00850 | 01142 | R,E,M |
| 00850 | 01143 | R,E,M |
| 00850 | 01144 | R,E,M |
| 00850 | 01145 | R,E,M |
| 00850 | 01146 | R,E,M |
| 00850 | 01147 | R,E,M |
| 00850 | 01148 | R,E,M |
| 00850 | 01149 | R,E,M |
| 00850 | 01153 | R,E,M |
| 00850 | 01250 | R |
| 00850 | 01251 | R |
| 00850 | 01252 | R,E |
| 00850 | 01253 | R |
| 00850 | 01254 | R |
| 00850 | 01255 | R |
| 00850 | 01256 | R |
| 00850 | 01257 | R |
| 00850 | 01275 | R |
| 00850 | 01280 | R |
| 00850 | 01281 | R |
| 00850 | 01283 | R |
| 00850 | 04953 | E |
| 00850 | 05348 | R,E |
| 00851 | 00259 | E |
| 00851 | 00367 | E |
| 00851 | 00423 | R |
| 00851 | 00500 | R |
| 00851 | 00875 | R |
| 00852 | 00037 | R,E,M |
| 00852 | 00256 | R,E |
| 00852 | 00259 | E |
| 00852 | 00273 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00852 | 00277 | R,E,M |
| 00852 | 00278 | R,E,M |
| 00852 | 00280 | R,E,M |
| 00852 | 00284 | R,E,M |
| 00852 | 00285 | R,E,M |
| 00852 | 00290 | R,E,M |
| 00852 | 00297 | R,E,M |
| 00852 | 00367 | E |
| 00852 | 00420 | R,E,M |
| 00852 | 00423 | R,E |
| 00852 | 00424 | R,E,M |
| 00852 | 00437 | R |
| 00852 | 00500 | R,E,M |
| 00852 | 00813 | R |
| 00852 | 00819 | R |
| 00852 | 00833 | R,E,M |
| 00852 | 00836 | R,M |
| 00852 | 00838 | R,E,M |
| 00852 | 00850 | R |
| 00852 | 00855 | R |
| 00852 | 00857 | R |
| 00852 | 00860 | R |
| 00852 | 00861 | R |
| 00852 | 00863 | R |
| 00852 | 00869 | R |
| 00852 | 00870 | R,E,M |
| 00852 | 00871 | R,E,M |
| 00852 | 00874 | R |
| 00852 | 00875 | R,E,M |
| 00852 | 00880 | R,E,M |
| 00852 | 00897 | R |
| 00852 | 00903 | R |
| 00852 | 00905 | R,E |
| 00852 | 00912 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00852 | 00916 | R |
| 00852 | 00920 | R |
| 00852 | 01025 | R,E,M |
| 00852 | 01026 | R,E,M |
| 00852 | 01027 | R,E,M |
| 00852 | 01040 | R,E |
| 00852 | 01041 | R,E |
| 00852 | 01042 | R |
| 00852 | 01043 | R,E |
| 00852 | 01047 | R,M |
| 00852 | 01088 | R |
| 00852 | 01097 | R,E |
| 00852 | 01153 | R,E,M |
| 00852 | 01250 | R |
| 00852 | 01252 | R |
| 00852 | 01282 | R |
| 00852 | 05346 | R,E |
| 00852 | 28709 | R,E,M |
| 00853 | 00367 | E |
| 00855 | 00037 | R,M |
| 00855 | 00259 | R |
| 00855 | 00273 | R,M |
| 00855 | 00277 | R,M |
| 00855 | 00278 | R,M |
| 00855 | 00280 | R,M |
| 00855 | 00284 | R,M |
| 00855 | 00285 | R,M |
| 00855 | 00290 | R,E,M |
| 00855 | 00297 | R,M |
| 00855 | 00367 | E |
| 00855 | 00437 | R |
| 00855 | 00500 | R,M |
| 00855 | 00819 | R |
| 00855 | 00833 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00855 | 00836 | R,M |
| 00855 | 00850 | R |
| 00855 | 00852 | R |
| 00855 | 00857 | R |
| 00855 | 00866 | E |
| 00855 | 00870 | R,M |
| 00855 | 00871 | R,M |
| 00855 | 00878 | R |
| 00855 | 00880 | R,M |
| 00855 | 00912 | R |
| 00855 | 00915 | R,E |
| 00855 | 01025 | R,E,M |
| 00855 | 01026 | R,M |
| 00855 | 01027 | R,E,M |
| 00855 | 01040 | R,E |
| 00855 | 01041 | R,E |
| 00855 | 01042 | R |
| 00855 | 01043 | R,E |
| 00855 | 01088 | R |
| 00855 | 01250 | R |
| 00855 | 01251 | R |
| 00855 | 01252 | R |
| 00855 | 01283 | R |
| 00855 | 05347 | R,E |
| 00856 | 00259 | E |
| 00856 | 00273 | E,L |
| 00856 | 00367 | E |
| 00856 | 00424 | R,M |
| 00856 | 00500 | R,M |
| 00856 | 00803 | R |
| 00856 | 00850 | R |
| 00856 | 00862 | R |
| 00856 | 00916 | R |
| 00856 | 01255 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00856 | 05351 | R,E |
| 00857 | 00037 | R,E |
| 00857 | 00256 | R,E |
| 00857 | 00259 | E |
| 00857 | 00273 | R,E |
| 00857 | 00277 | R,E |
| 00857 | 00278 | R,E |
| 00857 | 00280 | R,E |
| 00857 | 00284 | R,E |
| 00857 | 00285 | R,E |
| 00857 | 00290 | R,E |
| 00857 | 00297 | R,E |
| 00857 | 00367 | E |
| 00857 | 00420 | R,E |
| 00857 | 00423 | R,E |
| 00857 | 00424 | R,E |
| 00857 | 00437 | R |
| 00857 | 00500 | R,E |
| 00857 | 00813 | R |
| 00857 | 00819 | R |
| 00857 | 00833 | R,E |
| 00857 | 00836 | R |
| 00857 | 00838 | R,E |
| 00857 | 00850 | R |
| 00857 | 00852 | R |
| 00857 | 00855 | R |
| 00857 | 00860 | R |
| 00857 | 00861 | R |
| 00857 | 00863 | R |
| 00857 | 00869 | R |
| 00857 | 00870 | R,E |
| 00857 | 00871 | R,E |
| 00857 | 00874 | R |
| 00857 | 00875 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00857 | 00880 | R,E |
| 00857 | 00897 | R |
| 00857 | 00903 | R |
| 00857 | 00905 | R,E |
| 00857 | 00912 | R |
| 00857 | 00916 | R |
| 00857 | 00920 | R |
| 00857 | 01025 | R,E |
| 00857 | 01026 | R,E |
| 00857 | 01027 | R,E |
| 00857 | 01040 | R,E |
| 00857 | 01041 | R,E |
| 00857 | 01042 | R |
| 00857 | 01043 | R,E |
| 00857 | 01088 | R |
| 00857 | 01097 | R,E |
| 00857 | 01252 | R |
| 00857 | 01254 | R |
| 00857 | 01281 | R |
| 00857 | 01288 | R,E |
| 00857 | 05350 | R,E |
| 00857 | 28709 | R,E |
| 00858 | 00037 | R,E,M |
| 00858 | 00259 | E |
| 00858 | 00273 | R,E,M |
| 00858 | 00277 | R,E,M |
| 00858 | 00278 | R,E,M |
| 00858 | 00280 | R,E,M |
| 00858 | 00284 | R,E,M |
| 00858 | 00285 | R,E,M |
| 00858 | 00290 | E,L |
| 00858 | 00297 | R,E,M |
| 00858 | 00367 | E |
| 00858 | 00437 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00858 | 00500 | R,E,M |
| 00858 | 00808 | R,E |
| 00858 | 00819 | R,E |
| 00858 | 00850 | E |
| 00858 | 00860 | R,E |
| 00858 | 00861 | R,E |
| 00858 | 00865 | R,E |
| 00858 | 00871 | R,E,M |
| 00858 | 00872 | R,E |
| 00858 | 00901 | R,E |
| 00858 | 00902 | R,E |
| 00858 | 00923 | R,E |
| 00858 | 00924 | R,E,M |
| 00858 | 01027 | E,L |
| 00858 | 01047 | R,E,M |
| 00858 | 01051 | R,E |
| 00858 | 01140 | R,E,M |
| 00858 | 01141 | R,E,M |
| 00858 | 01142 | R,E,M |
| 00858 | 01143 | R,E,M |
| 00858 | 01144 | R,E,M |
| 00858 | 01145 | R,E,M |
| 00858 | 01146 | R,E,M |
| 00858 | 01147 | R,E,M |
| 00858 | 01148 | R,E,M |
| 00858 | 01149 | R,E,M |
| 00858 | 01153 | R,E,M |
| 00858 | 01154 | R,E,M |
| 00858 | 01155 | R,E,M |
| 00858 | 01156 | R,E,M |
| 00858 | 01157 | R,E,M |
| 00858 | 01160 | R,E,M |
| 00858 | 01161 | R,E |
| 00858 | 01162 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00858 | 01164 | R,E |
| 00858 | 01252 | R,E |
| 00858 | 01275 | R,E |
| 00858 | 04909 | R,E |
| 00858 | 04971 | R,E,M |
| 00858 | 05123 | E,L |
| 00858 | 05210 | R,E |
| 00858 | 05348 | R,E |
| 00858 | 08482 | R,E,M |
| 00858 | 09044 | R,E |
| 00858 | 09049 | R,E |
| 00858 | 09061 | R,E |
| 00858 | 16804 | R,E,M |
| 00858 | 17248 | R,E |
| 00859 | 00808 | R,E |
| 00859 | 00872 | R,E |
| 00859 | 00901 | R,E |
| 00859 | 00902 | R,E |
| 00859 | 01153 | R,E,M |
| 00859 | 01154 | R,E,M |
| 00859 | 01155 | R,E,M |
| 00859 | 01156 | R,E,M |
| 00859 | 01157 | R,E,M |
| 00859 | 01160 | R,E,M |
| 00859 | 01161 | R,E |
| 00859 | 01162 | R,E |
| 00859 | 01164 | R,E |
| 00859 | 04909 | R,E |
| 00859 | 04971 | R,E,M |
| 00859 | 09044 | R,E |
| 00859 | 09049 | R,E |
| 00859 | 09061 | R,E |
| 00859 | 16804 | R,E,M |
| 00859 | 17248 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00860 | 00037 | R,E |
| 00860 | 00256 | R,E |
| 00860 | 00259 | E |
| 00860 | 00273 | R,E |
| 00860 | 00277 | R,E |
| 00860 | 00278 | R,E |
| 00860 | 00280 | R,E |
| 00860 | 00284 | R,E |
| 00860 | 00285 | R,E |
| 00860 | 00290 | R,E |
| 00860 | 00297 | R,E |
| 00860 | 00367 | E |
| 00860 | 00420 | R,E |
| 00860 | 00423 | R,E |
| 00860 | 00424 | R,E |
| 00860 | 00437 | R |
| 00860 | 00500 | R,E |
| 00860 | 00813 | R |
| 00860 | 00819 | R |
| 00860 | 00833 | R,E |
| 00860 | 00838 | R,E |
| 00860 | 00850 | R |
| 00860 | 00852 | R |
| 00860 | 00857 | R |
| 00860 | 00858 | R,E |
| 00860 | 00861 | R |
| 00860 | 00863 | R |
| 00860 | 00865 | R |
| 00860 | 00869 | R |
| 00860 | 00870 | R,E |
| 00860 | 00871 | R,E |
| 00860 | 00874 | R |
| 00860 | 00875 | R,E |
| 00860 | 00880 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00860 | 00897 | R |
| 00860 | 00903 | R |
| 00860 | 00905 | R,E |
| 00860 | 00912 | R |
| 00860 | 00916 | R |
| 00860 | 00920 | R |
| 00860 | 00923 | R,E |
| 00860 | 00924 | R,E |
| 00860 | 01025 | R,E |
| 00860 | 01026 | R,E |
| 00860 | 01027 | R,E |
| 00860 | 01041 | R |
| 00860 | 01042 | R |
| 00860 | 01043 | R |
| 00860 | 01097 | R,E |
| 00860 | 01140 | R,E |
| 00860 | 01145 | R,E |
| 00860 | 01146 | R,E |
| 00860 | 01148 | R,E |
| 00860 | 01252 | R |
| 00860 | 05348 | R,E |
| 00860 | 28709 | R,E |
| 00861 | 00037 | R,E,M |
| 00861 | 00256 | R,E |
| 00861 | 00259 | E |
| 00861 | 00273 | R,E,M |
| 00861 | 00277 | R,E,M |
| 00861 | 00278 | R,E,M |
| 00861 | 00280 | R,E,M |
| 00861 | 00284 | R,E,M |
| 00861 | 00285 | R,E,M |
| 00861 | 00290 | R,E,M |
| 00861 | 00297 | R,E,M |
| 00861 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00861 | 00420 | R,E,M |
| 00861 | 00423 | R,E |
| 00861 | 00424 | R,E,M |
| 00861 | 00437 | R |
| 00861 | 00500 | R,E,M |
| 00861 | 00813 | R |
| 00861 | 00819 | R |
| 00861 | 00833 | R,E,M |
| 00861 | 00838 | R,E,M |
| 00861 | 00850 | R |
| 00861 | 00852 | R |
| 00861 | 00857 | R |
| 00861 | 00858 | R,E |
| 00861 | 00860 | R |
| 00861 | 00863 | R |
| 00861 | 00869 | R |
| 00861 | 00870 | R,E,M |
| 00861 | 00871 | R,E,M |
| 00861 | 00874 | R |
| 00861 | 00875 | R,E,M |
| 00861 | 00880 | R,E,M |
| 00861 | 00897 | R |
| 00861 | 00903 | R |
| 00861 | 00905 | R,E |
| 00861 | 00912 | R |
| 00861 | 00916 | R |
| 00861 | 00920 | R |
| 00861 | 00923 | R,E |
| 00861 | 00924 | R,E,M |
| 00861 | 01025 | R,E,M |
| 00861 | 01026 | R,E,M |
| 00861 | 01027 | R,E,M |
| 00861 | 01041 | R |
| 00861 | 01042 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00861 | 01043 | R |
| 00861 | 01097 | R,E |
| 00861 | 01148 | R,E,M |
| 00861 | 01149 | R,E,M |
| 00861 | 01252 | R |
| 00861 | 05348 | R,E |
| 00861 | 28709 | R,E,M |
| 00862 | 00037 | R,E,M |
| 00862 | 00256 | R,E |
| 00862 | 00259 | E |
| 00862 | 00273 | R,E,M |
| 00862 | 00277 | R,E,M |
| 00862 | 00278 | R,E,M |
| 00862 | 00280 | R,E,M |
| 00862 | 00284 | R,E,M |
| 00862 | 00285 | R,E,M |
| 00862 | 00290 | R,E,M |
| 00862 | 00297 | R,E,M |
| 00862 | 00367 | E |
| 00862 | 00420 | R,E,M |
| 00862 | 00423 | R,E |
| 00862 | 00424 | R,E,M |
| 00862 | 00437 | R |
| 00862 | 00500 | R,E,M |
| 00862 | 00803 | R,E |
| 00862 | 00833 | R,E,M |
| 00862 | 00838 | R,E,M |
| 00862 | 00850 | R |
| 00862 | 00856 | R |
| 00862 | 00870 | R,E,M |
| 00862 | 00871 | R,E,M |
| 00862 | 00875 | R,E,M |
| 00862 | 00880 | R,E,M |
| 00862 | 00905 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00862 | 00916 | R,E |
| 00862 | 01025 | R,E,M |
| 00862 | 01026 | R,E,M |
| 00862 | 01027 | R,E,M |
| 00862 | 01097 | R,E |
| 00862 | 01252 | R |
| 00862 | 01255 | R,E |
| 00862 | 05351 | R,E |
| 00862 | 12712 | R,E,M |
| 00862 | 28709 | R,E,M |
| 00863 | 00037 | R,E |
| 00863 | 00256 | R,E |
| 00863 | 00259 | E |
| 00863 | 00273 | R,E |
| 00863 | 00277 | R,E |
| 00863 | 00278 | R,E |
| 00863 | 00280 | R,E |
| 00863 | 00284 | R,E |
| 00863 | 00285 | R,E |
| 00863 | 00290 | R,E |
| 00863 | 00297 | R,E |
| 00863 | 00367 | E |
| 00863 | 00420 | R,E |
| 00863 | 00423 | R,E |
| 00863 | 00424 | R,E |
| 00863 | 00437 | R |
| 00863 | 00500 | R,E |
| 00863 | 00813 | R |
| 00863 | 00819 | R |
| 00863 | 00833 | R,E |
| 00863 | 00838 | R,E |
| 00863 | 00850 | R |
| 00863 | 00852 | R |
| 00863 | 00857 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00863 | 00860 | R |
| 00863 | 00861 | R |
| 00863 | 00865 | R |
| 00863 | 00869 | R |
| 00863 | 00870 | R,E |
| 00863 | 00871 | R,E |
| 00863 | 00874 | R |
| 00863 | 00875 | R,E |
| 00863 | 00880 | R,E |
| 00863 | 00897 | R |
| 00863 | 00903 | R |
| 00863 | 00905 | R,E |
| 00863 | 00912 | R |
| 00863 | 00916 | R |
| 00863 | 00920 | R |
| 00863 | 00923 | R,E |
| 00863 | 01025 | R,E |
| 00863 | 01026 | R,E |
| 00863 | 01027 | R,E |
| 00863 | 01041 | R |
| 00863 | 01042 | R |
| 00863 | 01043 | R |
| 00863 | 01051 | R |
| 00863 | 01097 | R,E |
| 00863 | 01140 | R,E |
| 00863 | 01141 | R,E |
| 00863 | 01142 | R,E |
| 00863 | 01143 | R,E |
| 00863 | 01144 | R,E |
| 00863 | 01145 | R,E |
| 00863 | 01146 | R,E |
| 00863 | 01147 | R,E |
| 00863 | 01148 | R,E |
| 00863 | 01149 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00863 | 01252 | R |
| 00863 | 01275 | R |
| 00863 | 05348 | R,E |
| 00863 | 28709 | R,E |
| 00864 | 00037 | R,E,M |
| 00864 | 00256 | R,E |
| 00864 | 00259 | E |
| 00864 | 00273 | R,E,M |
| 00864 | 00277 | R,E,M |
| 00864 | 00278 | R,E,M |
| 00864 | 00280 | R,E,M |
| 00864 | 00284 | R,E,M |
| 00864 | 00285 | R,E,M |
| 00864 | 00290 | R,E,M |
| 00864 | 00297 | R,E,M |
| 00864 | 00367 | E |
| 00864 | 00420 | R,E,M |
| 00864 | 00423 | R,E |
| 00864 | 00424 | R,E,M |
| 00864 | 00425 | C,L |
| 00864 | 00500 | R,E,M |
| 00864 | 00720 | C |
| 00864 | 00819 | R |
| 00864 | 00833 | R,E,M |
| 00864 | 00838 | R,E,M |
| 00864 | 00850 | R |
| 00864 | 00870 | R,E,M |
| 00864 | 00871 | R,E,M |
| 00864 | 00875 | R,E,M |
| 00864 | 00880 | R,E,M |
| 00864 | 00905 | R,E |
| 00864 | 00918 | R |
| 00864 | 01008 | R |
| 00864 | 01025 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00864 | 01026 | R,E,M |
| 00864 | 01027 | R,E,M |
| 00864 | 01046 | C |
| 00864 | 01089 | E,C |
| 00864 | 01097 | R,E |
| 00864 | 01127 | R |
| 00864 | 01252 | R |
| 00864 | 01256 | E |
| 00864 | 05352 | E |
| 00864 | 28709 | R,E,M |
| 00865 | 00037 | R,E |
| 00865 | 00256 | R,E |
| 00865 | 00259 | E |
| 00865 | 00273 | R,E |
| 00865 | 00277 | R,E |
| 00865 | 00278 | R,E |
| 00865 | 00280 | R,E |
| 00865 | 00284 | R,E |
| 00865 | 00285 | R,E |
| 00865 | 00290 | R,E |
| 00865 | 00297 | R,E |
| 00865 | 00367 | E |
| 00865 | 00420 | R,E |
| 00865 | 00423 | R,E |
| 00865 | 00424 | R,E |
| 00865 | 00437 | R |
| 00865 | 00500 | R,E |
| 00865 | 00819 | R |
| 00865 | 00833 | R,E |
| 00865 | 00838 | R,E |
| 00865 | 00850 | R |
| 00865 | 00858 | R,E |
| 00865 | 00860 | R |
| 00865 | 00863 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00865 | 00870 | R,E |
| 00865 | 00871 | R,E |
| 00865 | 00875 | R,E |
| 00865 | 00880 | R,E |
| 00865 | 00905 | R,E |
| 00865 | 00923 | R,E |
| 00865 | 00924 | R,E |
| 00865 | 01025 | R,E |
| 00865 | 01026 | R,E |
| 00865 | 01027 | R,E |
| 00865 | 01097 | R,E |
| 00865 | 01142 | R,E |
| 00865 | 01143 | R,E |
| 00865 | 01148 | R,E |
| 00865 | 01252 | R |
| 00865 | 05348 | R,E |
| 00865 | 28709 | R,E |
| 00866 | 00037 | R,M |
| 00866 | 00256 | E,C |
| 00866 | 00367 | E |
| 00866 | 00437 | R |
| 00866 | 00500 | E,L |
| 00866 | 00819 | R |
| 00866 | 00850 | R |
| 00866 | 00855 | E |
| 00866 | 00870 | R,M |
| 00866 | 00878 | R |
| 00866 | 00880 | E,L |
| 00866 | 00915 | E |
| 00866 | 01025 | R,E,M |
| 00866 | 01251 | R |
| 00866 | 01252 | R |
| 00866 | 01283 | R |
| 00866 | 05347 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00867 | 00259 | E |
| 00867 | 00424 | R,M |
| 00867 | 00916 | R,E |
| 00867 | 01148 | R,E,M |
| 00867 | 01153 | R,E,M |
| 00867 | 01154 | R,E,M |
| 00867 | 01155 | R,E,M |
| 00867 | 01160 | R,E,M |
| 00867 | 04899 | R,E |
| 00867 | 04971 | R,E,M |
| 00867 | 05012 | R,E |
| 00867 | 05351 | R,E |
| 00867 | 09048 | R,E |
| 00867 | 12712 | R,E,M |
| 00867 | 16804 | R,E,M |
| 00868 | 00367 | E |
| 00868 | 00918 | R |
| 00868 | 01006 | R |
| 00869 | 00037 | R,M |
| 00869 | 00256 | R |
| 00869 | 00259 | E |
| 00869 | 00273 | R,M |
| 00869 | 00277 | R,M |
| 00869 | 00278 | R,M |
| 00869 | 00280 | R,M |
| 00869 | 00284 | R,M |
| 00869 | 00285 | R,M |
| 00869 | 00297 | R,M |
| 00869 | 00367 | E |
| 00869 | 00423 | R |
| 00869 | 00437 | R |
| 00869 | 00500 | R,E,M |
| 00869 | 00737 | R,E |
| 00869 | 00813 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00869 | 00819 | R |
| 00869 | 00838 | R,M |
| 00869 | 00850 | R |
| 00869 | 00852 | R |
| 00869 | 00857 | R |
| 00869 | 00860 | R |
| 00869 | 00861 | R |
| 00869 | 00863 | R |
| 00869 | 00870 | R,M |
| 00869 | 00871 | R,M |
| 00869 | 00874 | R |
| 00869 | 00875 | R,E,M |
| 00869 | 00880 | R,M |
| 00869 | 00897 | R |
| 00869 | 00903 | R |
| 00869 | 00912 | R |
| 00869 | 00916 | R |
| 00869 | 00920 | R |
| 00869 | 01025 | R,M |
| 00869 | 01026 | R,M |
| 00869 | 01027 | R,M |
| 00869 | 01041 | R |
| 00869 | 01042 | R |
| 00869 | 01043 | R |
| 00869 | 01252 | R |
| 00869 | 01253 | R |
| 00869 | 01254 | R |
| 00869 | 01280 | R |
| 00869 | 01287 | R,E |
| 00869 | 05349 | R,E |
| 00870 | 00037 | R,E |
| 00870 | 00256 | R,E |
| 00870 | 00273 | R |
| 00870 | 00277 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00870 | 00278 | R |
| 00870 | 00280 | R |
| 00870 | 00284 | R |
| 00870 | 00285 | R |
| 00870 | 00290 | R,E |
| 00870 | 00297 | R |
| 00870 | 00367 | E |
| 00870 | 00423 | R |
| 00870 | 00437 | R,E,M |
| 00870 | 00500 | R,E |
| 00870 | 00737 | R |
| 00870 | 00775 | R,E |
| 00870 | 00813 | R,L |
| 00870 | 00819 | R,L |
| 00870 | 00833 | R,E |
| 00870 | 00836 | R |
| 00870 | 00838 | R |
| 00870 | 00850 | R,E,M |
| 00870 | 00852 | R,E,M |
| 00870 | 00855 | R,M |
| 00870 | 00857 | R,E |
| 00870 | 00860 | R,E |
| 00870 | 00861 | R,E,M |
| 00870 | 00862 | R,E,M |
| 00870 | 00863 | R,E |
| 00870 | 00864 | R,E,M |
| 00870 | 00865 | R,E |
| 00870 | 00866 | R,M |
| 00870 | 00869 | R,M |
| 00870 | 00871 | R |
| 00870 | 00874 | R,M |
| 00870 | 00875 | R |
| 00870 | 00880 | R |
| 00870 | 00897 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00870 | 00903 | R |
| 00870 | 00912 | R,L |
| 00870 | 00915 | R,L |
| 00870 | 00916 | R,L |
| 00870 | 00920 | R,L |
| 00870 | 01009 | E |
| 00870 | 01025 | R |
| 00870 | 01026 | R |
| 00870 | 01027 | R,E |
| 00870 | 01040 | R,E |
| 00870 | 01041 | R,E |
| 00870 | 01042 | R |
| 00870 | 01043 | R,E |
| 00870 | 01047 | R |
| 00870 | 01051 | E |
| 00870 | 01088 | R,L |
| 00870 | 01112 | R |
| 00870 | 01122 | R |
| 00870 | 01147 | R,E |
| 00870 | 01250 | R,E,L |
| 00870 | 01252 | R,L |
| 00870 | 01282 | R |
| 00870 | 05346 | R,E,L |
| 00870 | 09044 | R,E,L |
| 00871 | 00037 | R,E |
| 00871 | 00256 | R |
| 00871 | 00273 | R |
| 00871 | 00277 | R |
| 00871 | 00278 | R |
| 00871 | 00280 | R |
| 00871 | 00284 | R |
| 00871 | 00285 | R |
| 00871 | 00290 | R,E |
| 00871 | 00297 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00871 | 00367 | E |
| 00871 | 00423 | R |
| 00871 | 00437 | R,E,M |
| 00871 | 00500 | R,E |
| 00871 | 00737 | R |
| 00871 | 00775 | R |
| 00871 | 00813 | R,L |
| 00871 | 00819 | R,L |
| 00871 | 00833 | R,E |
| 00871 | 00836 | R,E |
| 00871 | 00838 | E |
| 00871 | 00850 | R,E,M |
| 00871 | 00852 | R,E,M |
| 00871 | 00855 | R,M |
| 00871 | 00857 | R,E |
| 00871 | 00858 | R,E,M |
| 00871 | 00860 | R,E |
| 00871 | 00861 | R,E,M |
| 00871 | 00862 | R,E,M |
| 00871 | 00863 | R,E |
| 00871 | 00864 | R,E,M |
| 00871 | 00865 | R,E |
| 00871 | 00869 | R,M |
| 00871 | 00870 | R |
| 00871 | 00874 | R,M |
| 00871 | 00875 | R |
| 00871 | 00880 | R |
| 00871 | 00897 | R |
| 00871 | 00903 | R |
| 00871 | 00912 | R,L |
| 00871 | 00916 | R,L |
| 00871 | 00920 | R,L |
| 00871 | 00923 | R,E,L |
| 00871 | 00924 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00871 | 01009 | E |
| 00871 | 01025 | R |
| 00871 | 01026 | R |
| 00871 | 01027 | R,E |
| 00871 | 01040 | R,E |
| 00871 | 01041 | R,E |
| 00871 | 01042 | R |
| 00871 | 01043 | R,E |
| 00871 | 01047 | R |
| 00871 | 01051 | R,E |
| 00871 | 01088 | R,L |
| 00871 | 01112 | R |
| 00871 | 01122 | R |
| 00871 | 01140 | R,E |
| 00871 | 01141 | R,E |
| 00871 | 01142 | R,E |
| 00871 | 01143 | R,E |
| 00871 | 01144 | R,E |
| 00871 | 01145 | R,E |
| 00871 | 01146 | R,E |
| 00871 | 01147 | R,E |
| 00871 | 01148 | R,E |
| 00871 | 01149 | E |
| 00871 | 01252 | R,E,L |
| 00871 | 01275 | R |
| 00871 | 05348 | R,E,L |
| 00872 | 00259 | E |
| 00872 | 00808 | R,E |
| 00872 | 00858 | R,E |
| 00872 | 00859 | R,E |
| 00872 | 00923 | R,E |
| 00872 | 00924 | R,E,M |
| 00872 | 01025 | R,E,M |
| 00872 | 01140 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00872 | 01141 | R,E,M |
| 00872 | 01142 | R,E,M |
| 00872 | 01143 | R,E,M |
| 00872 | 01144 | R,E,M |
| 00872 | 01145 | R,E,M |
| 00872 | 01146 | R,E,M |
| 00872 | 01147 | R,E,M |
| 00872 | 01148 | R,E,M |
| 00872 | 01149 | R,E,M |
| 00872 | 01153 | R,E,M |
| 00872 | 01154 | R,E,M |
| 00872 | 01155 | R,E,M |
| 00872 | 05346 | R,E |
| 00872 | 05347 | R,E |
| 00872 | 05348 | R,E |
| 00872 | 09044 | R,E |
| 00872 | 09049 | R,E |
| 00874 | 00037 | R,E,M |
| 00874 | 00259 | E |
| 00874 | 00273 | R,M |
| 00874 | 00277 | R,M |
| 00874 | 00278 | R,M |
| 00874 | 00280 | R,M |
| 00874 | 00284 | R,M |
| 00874 | 00285 | R,M |
| 00874 | 00297 | R,M |
| 00874 | 00367 | E |
| 00874 | 00423 | R |
| 00874 | 00437 | R |
| 00874 | 00813 | R |
| 00874 | 00819 | R |
| 00874 | 00838 | R,E,M |
| 00874 | 00850 | R |
| 00874 | 00852 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00874 | 00857 | R |
| 00874 | 00860 | R |
| 00874 | 00861 | R |
| 00874 | 00863 | R |
| 00874 | 00869 | R |
| 00874 | 00870 | R,M |
| 00874 | 00871 | R,M |
| 00874 | 00875 | R,M |
| 00874 | 00880 | R,M |
| 00874 | 00897 | R |
| 00874 | 00903 | R |
| 00874 | 00912 | R |
| 00874 | 00916 | R |
| 00874 | 00920 | R |
| 00874 | 01025 | R,M |
| 00874 | 01026 | R,M |
| 00874 | 01027 | R,M |
| 00874 | 01041 | R |
| 00874 | 01042 | R |
| 00874 | 01043 | R |
| 00874 | 01252 | E |
| 00874 | 04970 | E |
| 00875 | 00037 | R,E |
| 00875 | 00256 | R |
| 00875 | 00273 | R |
| 00875 | 00277 | R |
| 00875 | 00278 | R |
| 00875 | 00280 | R |
| 00875 | 00284 | R |
| 00875 | 00285 | R |
| 00875 | 00297 | R |
| 00875 | 00367 | E |
| 00875 | 00423 | R |
| 00875 | 00437 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00875 | 00500 | R,E |
| 00875 | 00737 | R,E |
| 00875 | 00775 | R,E |
| 00875 | 00813 | R,L |
| 00875 | 00819 | R,L |
| 00875 | 00836 | R,E |
| 00875 | 00838 | R |
| 00875 | 00850 | R,M |
| 00875 | 00851 | R |
| 00875 | 00852 | R,E,M |
| 00875 | 00857 | R,E |
| 00875 | 00860 | R,E |
| 00875 | 00861 | R,E,M |
| 00875 | 00862 | R,E,M |
| 00875 | 00863 | R,E |
| 00875 | 00864 | R,E,M |
| 00875 | 00865 | R,E |
| 00875 | 00869 | R,E,M |
| 00875 | 00870 | R |
| 00875 | 00871 | R |
| 00875 | 00874 | R,M |
| 00875 | 00880 | R |
| 00875 | 00897 | R |
| 00875 | 00903 | R |
| 00875 | 00912 | R,L |
| 00875 | 00916 | R,L |
| 00875 | 00920 | R,L |
| 00875 | 01009 | E |
| 00875 | 01025 | R |
| 00875 | 01026 | R |
| 00875 | 01027 | R |
| 00875 | 01041 | R |
| 00875 | 01042 | R |
| 00875 | 01043 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00875 | 01047 | R |
| 00875 | 01051 | E |
| 00875 | 01088 | R,L |
| 00875 | 01112 | R |
| 00875 | 01122 | R |
| 00875 | 01252 | R,L |
| 00875 | 01253 | R,E,L |
| 00875 | 01280 | R |
| 00875 | 01287 | R,E |
| 00875 | 04909 | R,E,L |
| 00875 | 05349 | R,E,L |
| 00875 | 09061 | R,E |
| 00878 | 00855 | R |
| 00878 | 00866 | R |
| 00878 | 00880 | R,E |
| 00878 | 00915 | R |
| 00878 | 01025 | R,E |
| 00878 | 01131 | R,E |
| 00878 | 01251 | R |
| 00878 | 01283 | R,E |
| 00878 | 05347 | R,E |
| 00880 | 00037 | R,E |
| 00880 | 00256 | R |
| 00880 | 00273 | R |
| 00880 | 00277 | R |
| 00880 | 00278 | R |
| 00880 | 00280 | R |
| 00880 | 00284 | R |
| 00880 | 00285 | R |
| 00880 | 00297 | R |
| 00880 | 00367 | E |
| 00880 | 00423 | R |
| 00880 | 00437 | R,E,M |
| 00880 | 00500 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00880 | 00737 | R |
| 00880 | 00775 | R |
| 00880 | 00813 | R,L |
| 00880 | 00819 | R,L |
| 00880 | 00838 | R |
| 00880 | 00850 | R,E,M |
| 00880 | 00852 | R,E,M |
| 00880 | 00855 | R,M |
| 00880 | 00857 | R,E |
| 00880 | 00860 | R,E |
| 00880 | 00861 | R,E,M |
| 00880 | 00862 | R,E,M |
| 00880 | 00863 | R,E |
| 00880 | 00864 | R,E,M |
| 00880 | 00865 | R,E |
| 00880 | 00866 | E,L |
| 00880 | 00869 | R,M |
| 00880 | 00870 | R |
| 00880 | 00871 | R |
| 00880 | 00874 | R,M |
| 00880 | 00875 | R |
| 00880 | 00878 | R,E |
| 00880 | 00897 | R |
| 00880 | 00903 | R |
| 00880 | 00912 | R,L |
| 00880 | 00915 | R,L |
| 00880 | 00916 | R,L |
| 00880 | 00920 | R,L |
| 00880 | 01009 | E |
| 00880 | 01025 | R,E |
| 00880 | 01026 | R |
| 00880 | 01027 | R |
| 00880 | 01041 | R |
| 00880 | 01042 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00880 | 01043 | R |
| 00880 | 01051 | E |
| 00880 | 01112 | R |
| 00880 | 01122 | R |
| 00880 | 01251 | R,E,L |
| 00880 | 01252 | R,L |
| 00880 | 01283 | R |
| 00880 | 05347 | R,E,L |
| 00891 | 00367 | E |
| 00891 | 00500 | E |
| 00891 | 00833 | E |
| 00891 | 01088 | E |
| 00895 | 00290 | E |
| 00895 | 00367 | E |
| 00895 | 00500 | E |
| 00895 | 01027 | E |
| 00895 | 01041 | E |
| 00896 | 00290 | E |
| 00896 | 00367 | E |
| 00896 | 01027 | E |
| 00896 | 01041 | E |
| 00897 | 00037 | R,E |
| 00897 | 00273 | R |
| 00897 | 00277 | R |
| 00897 | 00278 | R |
| 00897 | 00280 | R |
| 00897 | 00284 | R |
| 00897 | 00285 | R |
| 00897 | 00290 | E |
| 00897 | 00297 | R |
| 00897 | 00367 | E |
| 00897 | 00423 | R |
| 00897 | 00437 | R,E |
| 00897 | 00500 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00897 | 00813 | R |
| 00897 | 00819 | R |
| 00897 | 00838 | R |
| 00897 | 00850 | R,E |
| 00897 | 00852 | R |
| 00897 | 00857 | R |
| 00897 | 00860 | R |
| 00897 | 00861 | R |
| 00897 | 00863 | R |
| 00897 | 00869 | R |
| 00897 | 00870 | R |
| 00897 | 00871 | R |
| 00897 | 00874 | R |
| 00897 | 00875 | R |
| 00897 | 00880 | R |
| 00897 | 00903 | R |
| 00897 | 00912 | R |
| 00897 | 00916 | R |
| 00897 | 00920 | R |
| 00897 | 01025 | R |
| 00897 | 01026 | R |
| 00897 | 01027 | E |
| 00897 | 01041 | E |
| 00897 | 01042 | R |
| 00897 | 01043 | R |
| 00897 | 01252 | E |
| 00899 | 00259 | E |
| 00901 | 00037 | R,E,L |
| 00901 | 00259 | E |
| 00901 | 00500 | R,E,L |
| 00901 | 00858 | R,E |
| 00901 | 00859 | R,E |
| 00901 | 00902 | R,E |
| 00901 | 00923 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00901 | 00924 | R,E,L |
| 00901 | 01140 | R,E,L |
| 00901 | 01148 | R,E,L |
| 00901 | 01156 | R,E,L |
| 00901 | 01157 | R,E,L |
| 00901 | 05348 | R,E |
| 00901 | 05353 | R,E |
| 00902 | 00037 | R,E,L |
| 00902 | 00259 | E |
| 00902 | 00500 | R,E,L |
| 00902 | 00858 | R,E |
| 00902 | 00859 | R,E |
| 00902 | 00901 | R,E |
| 00902 | 00923 | R,E |
| 00902 | 00924 | R,E,L |
| 00902 | 01140 | R,E,L |
| 00902 | 01148 | R,E,L |
| 00902 | 01156 | R,E,L |
| 00902 | 01157 | R,E,L |
| 00902 | 05348 | R,E |
| 00902 | 05353 | R,E |
| 00903 | 00037 | R |
| 00903 | 00273 | R |
| 00903 | 00277 | R |
| 00903 | 00278 | R |
| 00903 | 00280 | R |
| 00903 | 00284 | R |
| 00903 | 00285 | R |
| 00903 | 00297 | R |
| 00903 | 00367 | E |
| 00903 | 00423 | R |
| 00903 | 00437 | R |
| 00903 | 00500 | E |
| 00903 | 00813 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00903 | 00819 | R |
| 00903 | 00836 | E |
| 00903 | 00838 | R |
| 00903 | 00850 | R |
| 00903 | 00852 | R |
| 00903 | 00857 | R |
| 00903 | 00860 | R |
| 00903 | 00861 | R |
| 00903 | 00863 | R |
| 00903 | 00869 | R |
| 00903 | 00870 | R |
| 00903 | 00871 | R |
| 00903 | 00874 | R |
| 00903 | 00875 | R |
| 00903 | 00880 | R |
| 00903 | 00897 | R |
| 00903 | 00912 | R |
| 00903 | 00916 | R |
| 00903 | 00920 | R |
| 00903 | 01025 | R |
| 00903 | 01026 | R |
| 00903 | 01027 | R |
| 00903 | 01041 | R |
| 00903 | 01042 | R |
| 00903 | 01043 | R |
| 00903 | 01115 | E |
| 00903 | 01252 | E |
| 00904 | 00037 | E,L |
| 00904 | 00367 | E |
| 00904 | 00500 | E,L |
| 00904 | 01114 | E |
| 00905 | 00037 | R,E |
| 00905 | 00256 | R |
| 00905 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00905 | 00437 | R,E |
| 00905 | 00500 | R,E |
| 00905 | 00737 | R |
| 00905 | 00775 | R,E |
| 00905 | 00819 | R |
| 00905 | 00850 | R,E |
| 00905 | 00852 | R,E |
| 00905 | 00857 | R,E |
| 00905 | 00860 | R,E |
| 00905 | 00861 | R,E |
| 00905 | 00862 | R,E |
| 00905 | 00863 | R,E |
| 00905 | 00864 | R,E |
| 00905 | 00865 | R,E |
| 00905 | 00920 | R |
| 00905 | 01026 | R |
| 00905 | 01051 | E |
| 00905 | 01112 | R |
| 00905 | 01122 | R |
| 00905 | 01252 | R |
| 00905 | 01254 | R,E |
| 00905 | 01281 | R |
| 00912 | 00037 | R,L |
| 00912 | 00273 | R,L |
| 00912 | 00277 | R,L |
| 00912 | 00278 | R,L |
| 00912 | 00280 | R,L |
| 00912 | 00284 | R,L |
| 00912 | 00285 | R,L |
| 00912 | 00297 | R,L |
| 00912 | 00367 | E |
| 00912 | 00423 | R |
| 00912 | 00437 | R |
| 00912 | 00500 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00912 | 00813 | R |
| 00912 | 00819 | R |
| 00912 | 00838 | R,L |
| 00912 | 00850 | R |
| 00912 | 00852 | R,E |
| 00912 | 00855 | R |
| 00912 | 00857 | R |
| 00912 | 00860 | R |
| 00912 | 00861 | R |
| 00912 | 00863 | R |
| 00912 | 00869 | R |
| 00912 | 00870 | R,L |
| 00912 | 00871 | R,L |
| 00912 | 00874 | R |
| 00912 | 00875 | R,L |
| 00912 | 00880 | R,L |
| 00912 | 00897 | R |
| 00912 | 00903 | R |
| 00912 | 00916 | R |
| 00912 | 00920 | R |
| 00912 | 01025 | R,L |
| 00912 | 01026 | R,L |
| 00912 | 01027 | R,L |
| 00912 | 01041 | R |
| 00912 | 01042 | R |
| 00912 | 01043 | R |
| 00912 | 01047 | R,L |
| 00912 | 01148 | E,L |
| 00912 | 01153 | R,E,L |
| 00912 | 01250 | R |
| 00912 | 01252 | R |
| 00912 | 01282 | R |
| 00912 | 05346 | R,E |
| 00914 | 00037 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00914 | 00437 | R |
| 00914 | 00500 | R,L |
| 00914 | 00819 | R |
| 00914 | 00850 | R |
| 00914 | 01112 | R,E,L |
| 00914 | 01122 | R,E,L |
| 00914 | 01252 | R |
| 00914 | 01257 | R |
| 00915 | 00037 | R,L |
| 00915 | 00259 | R |
| 00915 | 00367 | E |
| 00915 | 00437 | R |
| 00915 | 00500 | R,L |
| 00915 | 00819 | R |
| 00915 | 00850 | R |
| 00915 | 00855 | R,E |
| 00915 | 00866 | E |
| 00915 | 00870 | R,L |
| 00915 | 00878 | R |
| 00915 | 00880 | R,L |
| 00915 | 01025 | R,E,L |
| 00915 | 01131 | R |
| 00915 | 01154 | R,E,L |
| 00915 | 01167 | R,E |
| 00915 | 01251 | R |
| 00915 | 01252 | R |
| 00915 | 01283 | R |
| 00915 | 05347 | R,E |
| 00916 | 00037 | R,L |
| 00916 | 00273 | R,L |
| 00916 | 00277 | R,L |
| 00916 | 00278 | R,L |
| 00916 | 00280 | R,L |
| 00916 | 00284 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 00916 | 00285 | R,L |
| 00916 | 00297 | R,L |
| 00916 | 00367 | E |
| 00916 | 00423 | R |
| 00916 | 00424 | R,E,L |
| 00916 | 00437 | R |
| 00916 | 00500 | R,L |
| 00916 | 00803 | R,E |
| 00916 | 00813 | R |
| 00916 | 00819 | R |
| 00916 | 00838 | R,L |
| 00916 | 00850 | R |
| 00916 | 00852 | R |
| 00916 | 00856 | R |
| 00916 | 00857 | R |
| 00916 | 00860 | R |
| 00916 | 00861 | R |
| 00916 | 00862 | R,E |
| 00916 | 00863 | R |
| 00916 | 00867 | R,E |
| 00916 | 00869 | R |
| 00916 | 00870 | R,L |
| 00916 | 00871 | R,L |
| 00916 | 00874 | R |
| 00916 | 00875 | R,L |
| 00916 | 00880 | R,L |
| 00916 | 00897 | R |
| 00916 | 00903 | R |
| 00916 | 00912 | R |
| 00916 | 00920 | R |
| 00916 | 01025 | R,L |
| 00916 | 01026 | R,L |
| 00916 | 01027 | R,L |
| 00916 | 01041 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00916 | 01042 | R |
| 00916 | 01043 | R |
| 00916 | 01148 | E,L |
| 00916 | 01252 | R |
| 00916 | 01255 | R,E |
| 00916 | 05351 | R,E |
| 00916 | 09048 | R,E |
| 00916 | 12712 | R,E,L |
| 00918 | 00367 | E |
| 00918 | 00864 | R |
| 00918 | 00868 | R |
| 00918 | 01006 | R |
| 00920 | 00037 | R,L |
| 00920 | 00273 | R,L |
| 00920 | 00277 | R,L |
| 00920 | 00278 | R,L |
| 00920 | 00280 | R,L |
| 00920 | 00284 | R,L |
| 00920 | 00285 | R,L |
| 00920 | 00297 | R,L |
| 00920 | 00367 | E |
| 00920 | 00423 | R |
| 00920 | 00437 | R |
| 00920 | 00500 | R,L |
| 00920 | 00813 | R |
| 00920 | 00819 | R |
| 00920 | 00838 | R,L |
| 00920 | 00850 | R |
| 00920 | 00852 | R |
| 00920 | 00857 | R |
| 00920 | 00860 | R |
| 00920 | 00861 | R |
| 00920 | 00863 | R |
| 00920 | 00869 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00920 | 00870 | R,L |
| 00920 | 00871 | R,L |
| 00920 | 00874 | R |
| 00920 | 00875 | R,L |
| 00920 | 00880 | R,L |
| 00920 | 00897 | R |
| 00920 | 00903 | R |
| 00920 | 00905 | R |
| 00920 | 00912 | R |
| 00920 | 00916 | R |
| 00920 | 01025 | R,L |
| 00920 | 01026 | R,L |
| 00920 | 01148 | E,L |
| 00920 | 01155 | R,E,L |
| 00920 | 01252 | R |
| 00920 | 01254 | R |
| 00920 | 01281 | R |
| 00920 | 01288 | R,E |
| 00920 | 05350 | R,E |
| 00921 | 00037 | R,L |
| 00921 | 00367 | E |
| 00921 | 00437 | R |
| 00921 | 00500 | R,L |
| 00921 | 00819 | R |
| 00921 | 00850 | R |
| 00921 | 00922 | R |
| 00921 | 01112 | R,E,L |
| 00921 | 01122 | R,L |
| 00921 | 01252 | R |
| 00921 | 01257 | R,E |
| 00921 | 05353 | R,E |
| 00922 | 00037 | R,L |
| 00922 | 00367 | E |
| 00922 | 00437 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00922 | 00500 | R,L |
| 00922 | 00819 | R |
| 00922 | 00850 | R |
| 00922 | 00921 | R |
| 00922 | 01112 | R,L |
| 00922 | 01122 | R,E,L |
| 00922 | 01252 | R |
| 00922 | 01257 | R,E |
| 00922 | 05353 | R,E |
| 00923 | 00037 | R,E,L |
| 00923 | 00273 | R,E,L |
| 00923 | 00277 | R,E,L |
| 00923 | 00278 | R,E,L |
| 00923 | 00280 | R,E,L |
| 00923 | 00284 | R,E,L |
| 00923 | 00285 | R,E,L |
| 00923 | 00297 | R,E,L |
| 00923 | 00367 | E |
| 00923 | 00437 | R,E |
| 00923 | 00500 | R,E,L |
| 00923 | 00808 | R,E |
| 00923 | 00819 | E |
| 00923 | 00850 | R,E |
| 00923 | 00858 | R,E |
| 00923 | 00860 | R,E |
| 00923 | 00861 | R,E |
| 00923 | 00863 | R,E |
| 00923 | 00865 | R,E |
| 00923 | 00871 | R,E,L |
| 00923 | 00872 | R,E |
| 00923 | 00901 | R,E |
| 00923 | 00902 | R,E |
| 00923 | 00924 | R |
| 00923 | 01043 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00923 | 01047 | R,E,L |
| 00923 | 01051 | R,E |
| 00923 | 01140 | R,E,L |
| 00923 | 01141 | R,E,L |
| 00923 | 01142 | R,E,L |
| 00923 | 01143 | R,E,L |
| 00923 | 01144 | R,E,L |
| 00923 | 01145 | R,E,L |
| 00923 | 01146 | R,E,L |
| 00923 | 01147 | R,E,L |
| 00923 | 01148 | R,E,L |
| 00923 | 01149 | R,E,L |
| 00923 | 01153 | R,E,L |
| 00923 | 01154 | R,E,L |
| 00923 | 01155 | R,E,L |
| 00923 | 01156 | R,E,L |
| 00923 | 01157 | R,E,L |
| 00923 | 01158 | R,E,L |
| 00923 | 01160 | R,E,L |
| 00923 | 01161 | R,E |
| 00923 | 01162 | R,E |
| 00923 | 01164 | R,E |
| 00923 | 01252 | R,E |
| 00923 | 01275 | R,E |
| 00923 | 04909 | R,E |
| 00923 | 04971 | R,E,L |
| 00923 | 05210 | R,E |
| 00923 | 05348 | R,E |
| 00923 | 09044 | R,E |
| 00923 | 09049 | R,E |
| 00923 | 09061 | R,E |
| 00923 | 16804 | R,E,L |
| 00923 | 17248 | R,E |
| 00924 | 00037 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00924 | 00273 | R,E |
| 00924 | 00277 | R,E |
| 00924 | 00278 | R,E |
| 00924 | 00280 | R,E |
| 00924 | 00284 | R,E |
| 00924 | 00285 | R,E |
| 00924 | 00297 | R,E |
| 00924 | 00367 | E |
| 00924 | 00437 | R,E,M |
| 00924 | 00500 | R,E |
| 00924 | 00808 | R,E,M |
| 00924 | 00819 | R,E,L |
| 00924 | 00848 | R,E,M |
| 00924 | 00849 | R,E |
| 00924 | 00850 | R,E,M,P |
| 00924 | 00858 | R,E,M |
| 00924 | 00860 | R,E |
| 00924 | 00861 | R,E,M |
| 00924 | 00865 | R,E |
| 00924 | 00871 | R,E |
| 00924 | 00872 | R,E,M |
| 00924 | 00901 | R,E,L |
| 00924 | 00902 | R,E,L |
| 00924 | 00923 | R |
| 00924 | 01047 | R,E |
| 00924 | 01051 | R,E |
| 00924 | 01140 | R,E |
| 00924 | 01141 | R,E |
| 00924 | 01142 | R,E |
| 00924 | 01143 | R,E |
| 00924 | 01144 | R,E |
| 00924 | 01145 | R,E |
| 00924 | 01146 | R,E |
| 00924 | 01147 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00924 | 01148 | R,E |
| 00924 | 01149 | R,E |
| 00924 | 01153 | R,E |
| 00924 | 01154 | R,E |
| 00924 | 01155 | R,E |
| 00924 | 01156 | R,E |
| 00924 | 01157 | R,E |
| 00924 | 01160 | R,E |
| 00924 | 01161 | R,E,L |
| 00924 | 01162 | R,E |
| 00924 | 01163 | R,E |
| 00924 | 01164 | R,E |
| 00924 | 01252 | R,E,L |
| 00924 | 01275 | R,E |
| 00924 | 04909 | R,E,L |
| 00924 | 04971 | R,E |
| 00924 | 05348 | R,E,L |
| 00924 | 09044 | R,E,L |
| 00924 | 09049 | R,E |
| 00924 | 09061 | R,E |
| 00924 | 09238 | R,E,L |
| 00924 | 16804 | R,E |
| 00924 | 17248 | R,E,L |
| 00926 | 00834 | E |
| 00926 | 00951 | E |
| 00926 | 01362 | E |
| 00927 | 00835 | E |
| 00927 | 00947 | E |
| 00928 | 00837 | E |
| 00928 | 01380 | E |
| 00928 | 01385 | E |
| 00941 | 00300 | E |
| 00941 | 00301 | E |
| 00941 | 01351 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 00947 | 00835 | E |
| 00947 | 00927 | E |
| 00951 | 00834 | E |
| 00951 | 00926 | E |
| 00951 | 00971 | E |
| 00951 | 01362 | E |
| 00951 | 04930 | E |
| 00952 | 00300 | E |
| 00953 | 00300 | E |
| 00955 | 00300 | E |
| 00971 | 00834 | E |
| 00971 | 00951 | E |
| 00971 | 01362 | E |
| 01004 | 00367 | E |
| 01004 | 00500 | R |
| 01004 | 00819 | R |
| 01004 | 00850 | R |
| 01006 | 00367 | E |
| 01006 | 00868 | R |
| 01006 | 00918 | R |
| 01008 | 00367 | E |
| 01008 | 00420 | R |
| 01008 | 00864 | R |
| 01009 | 00037 | E |
| 01009 | 00273 | E |
| 01009 | 00277 | E |
| 01009 | 00278 | E |
| 01009 | 00280 | E |
| 01009 | 00284 | E |
| 01009 | 00290 | E |
| 01009 | 00297 | E |
| 01009 | 00367 | R |
| 01009 | 00423 | E |
| 01009 | 00500 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01009 | 00833 | E |
| 01009 | 00836 | E |
| 01009 | 00870 | E |
| 01009 | 00871 | E |
| 01009 | 00875 | E |
| 01009 | 00880 | E |
| 01009 | 01025 | E |
| 01009 | 01026 | E |
| 01010 | 00367 | E |
| 01010 | 00500 | E |
| 01011 | 00367 | E |
| 01011 | 00500 | E |
| 01012 | 00367 | E |
| 01012 | 00500 | E |
| 01013 | 00367 | E |
| 01013 | 00500 | E |
| 01013 | 01140 | E |
| 01014 | 00367 | E |
| 01014 | 00500 | E |
| 01015 | 00367 | E |
| 01015 | 00500 | E |
| 01016 | 00367 | E |
| 01016 | 00500 | E |
| 01017 | 00367 | E |
| 01017 | 00500 | E |
| 01018 | 00367 | E |
| 01018 | 00500 | E |
| 01019 | 00367 | E |
| 01019 | 00500 | E |
| 01020 | 00367 | E |
| 01020 | 00500 | E |
| 01021 | 00367 | E |
| 01021 | 00500 | E |
| 01023 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01023 | 00500 | E |
| 01025 | 00037 | R,E |
| 01025 | 00256 | R |
| 01025 | 00273 | R |
| 01025 | 00277 | R |
| 01025 | 00278 | R |
| 01025 | 00280 | R |
| 01025 | 00284 | R |
| 01025 | 00285 | R |
| 01025 | 00290 | R,E |
| 01025 | 00297 | R |
| 01025 | 00367 | E |
| 01025 | 00423 | R |
| 01025 | 00437 | R,E,M |
| 01025 | 00500 | R,E |
| 01025 | 00737 | R |
| 01025 | 00775 | R |
| 01025 | 00808 | R,E,M |
| 01025 | 00813 | R,L |
| 01025 | 00819 | R,L |
| 01025 | 00833 | R,E |
| 01025 | 00836 | R |
| 01025 | 00838 | R |
| 01025 | 00849 | R,E |
| 01025 | 00850 | R,E,M |
| 01025 | 00852 | R,E,M |
| 01025 | 00855 | R,E,M |
| 01025 | 00857 | R,E |
| 01025 | 00860 | R,E |
| 01025 | 00861 | R,E,M |
| 01025 | 00862 | R,E,M |
| 01025 | 00863 | R,E |
| 01025 | 00864 | R,E,M |
| 01025 | 00865 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01025 | 00866 | R,E,M |
| 01025 | 00869 | R,M |
| 01025 | 00870 | R |
| 01025 | 00871 | R |
| 01025 | 00872 | R,E,M |
| 01025 | 00874 | R,M |
| 01025 | 00875 | R |
| 01025 | 00878 | R,E |
| 01025 | 00880 | R,E |
| 01025 | 00897 | R |
| 01025 | 00903 | R |
| 01025 | 00912 | R,L |
| 01025 | 00915 | R,E,L |
| 01025 | 00916 | R,L |
| 01025 | 00920 | R,L |
| 01025 | 01009 | E |
| 01025 | 01026 | R |
| 01025 | 01027 | R,E |
| 01025 | 01040 | R,E |
| 01025 | 01041 | R,E |
| 01025 | 01042 | R |
| 01025 | 01043 | R,E |
| 01025 | 01051 | R |
| 01025 | 01088 | R,L |
| 01025 | 01112 | R |
| 01025 | 01122 | R |
| 01025 | 01131 | R |
| 01025 | 01167 | R,E |
| 01025 | 01251 | R,E,L |
| 01025 | 01252 | R,L |
| 01025 | 01283 | R |
| 01025 | 05347 | R,E,L |
| 01026 | 00037 | R,E |
| 01026 | 00256 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01026 | 00273 | R |
| 01026 | 00277 | R |
| 01026 | 00278 | R |
| 01026 | 00280 | R |
| 01026 | 00284 | R |
| 01026 | 00285 | R |
| 01026 | 00290 | R,E |
| 01026 | 00297 | R |
| 01026 | 00367 | E |
| 01026 | 00423 | R |
| 01026 | 00437 | R,E,M |
| 01026 | 00500 | R,E |
| 01026 | 00737 | R |
| 01026 | 00775 | R,E |
| 01026 | 00813 | R,L |
| 01026 | 00819 | R,L |
| 01026 | 00833 | R,E |
| 01026 | 00836 | R |
| 01026 | 00838 | R |
| 01026 | 00850 | R,E,M |
| 01026 | 00852 | R,E,M |
| 01026 | 00855 | R,M |
| 01026 | 00857 | R,E |
| 01026 | 00860 | R,E |
| 01026 | 00861 | R,E,M |
| 01026 | 00862 | R,E,M |
| 01026 | 00863 | R,E |
| 01026 | 00864 | R,E,M |
| 01026 | 00865 | R,E |
| 01026 | 00869 | R,M |
| 01026 | 00870 | R |
| 01026 | 00871 | R |
| 01026 | 00874 | R,M |
| 01026 | 00875 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01026 | 00880 | R |
| 01026 | 00897 | R |
| 01026 | 00903 | R |
| 01026 | 00905 | R |
| 01026 | 00912 | R,L |
| 01026 | 00916 | R,L |
| 01026 | 00920 | R,L |
| 01026 | 01009 | E |
| 01026 | 01025 | R |
| 01026 | 01027 | R,E |
| 01026 | 01040 | R,E |
| 01026 | 01041 | R,E |
| 01026 | 01042 | R |
| 01026 | 01043 | R,E |
| 01026 | 01047 | R |
| 01026 | 01088 | R,L |
| 01026 | 01112 | R |
| 01026 | 01122 | R |
| 01026 | 01252 | R,L |
| 01026 | 01254 | R,E |
| 01026 | 01281 | R |
| 01026 | 01288 | R,E |
| 01026 | 05350 | R,E,L |
| 01026 | 09049 | R |
| 01027 | 00037 | R,E |
| 01027 | 00256 | E |
| 01027 | 00273 | R,E |
| 01027 | 00277 | R,E |
| 01027 | 00278 | R,E |
| 01027 | 00280 | R,E |
| 01027 | 00284 | R,E |
| 01027 | 00285 | R,E |
| 01027 | 00290 | R |
| 01027 | 00297 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01027 | 00367 | E |
| 01027 | 00423 | R |
| 01027 | 00437 | R,E,M |
| 01027 | 00500 | R,E |
| 01027 | 00737 | E |
| 01027 | 00775 | E |
| 01027 | 00813 | R,L |
| 01027 | 00819 | R,E,L |
| 01027 | 00833 | R,E |
| 01027 | 00836 | R,E |
| 01027 | 00838 | R |
| 01027 | 00850 | R,E,M |
| 01027 | 00852 | R,E,M |
| 01027 | 00855 | R,E,M |
| 01027 | 00857 | R,E |
| 01027 | 00858 | E,L |
| 01027 | 00860 | R,E |
| 01027 | 00861 | R,E,M |
| 01027 | 00862 | R,E,M |
| 01027 | 00863 | R,E |
| 01027 | 00864 | R,E,M |
| 01027 | 00865 | R,E |
| 01027 | 00869 | R,M |
| 01027 | 00870 | R,E |
| 01027 | 00871 | R,E |
| 01027 | 00874 | R,M |
| 01027 | 00875 | R |
| 01027 | 00880 | R |
| 01027 | 00895 | E |
| 01027 | 00896 | E |
| 01027 | 00897 | E |
| 01027 | 00903 | R |
| 01027 | 00912 | R,L |
| 01027 | 00916 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01027 | 01025 | R,E |
| 01027 | 01026 | R,E |
| 01027 | 01040 | R,E |
| 01027 | 01041 | R,E |
| 01027 | 01042 | R |
| 01027 | 01043 | R,E |
| 01027 | 01047 | R |
| 01027 | 01088 | R,L |
| 01027 | 01112 | R |
| 01027 | 01122 | R |
| 01027 | 01148 | R,E |
| 01027 | 01252 | E,L |
| 01027 | 05348 | E,L |
| 01040 | 00037 | R,E |
| 01040 | 00273 | R,E |
| 01040 | 00277 | R,E |
| 01040 | 00278 | R,E |
| 01040 | 00280 | R,E |
| 01040 | 00284 | R,E |
| 01040 | 00285 | R,E |
| 01040 | 00290 | R,E |
| 01040 | 00297 | R,E |
| 01040 | 00367 | E |
| 01040 | 00437 | R,E |
| 01040 | 00500 | R,E |
| 01040 | 00833 | R,E |
| 01040 | 00836 | R |
| 01040 | 00850 | R,E |
| 01040 | 00852 | R,E |
| 01040 | 00855 | R,E |
| 01040 | 00857 | R,E |
| 01040 | 00870 | R,E |
| 01040 | 00871 | R,E |
| 01040 | 01025 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01040 | 01026 | R,E |
| 01040 | 01027 | R,E |
| 01040 | 01041 | R,E |
| 01040 | 01042 | R |
| 01040 | 01043 | R,E |
| 01040 | 01088 | R,E |
| 01041 | 00037 | R,E |
| 01041 | 00273 | R,E |
| 01041 | 00277 | R,E |
| 01041 | 00278 | R,E |
| 01041 | 00280 | R,E |
| 01041 | 00284 | R,E |
| 01041 | 00285 | R,E |
| 01041 | 00290 | R,E |
| 01041 | 00297 | R,E |
| 01041 | 00367 | E |
| 01041 | 00423 | R |
| 01041 | 00437 | R,E |
| 01041 | 00500 | R,E |
| 01041 | 00813 | R |
| 01041 | 00819 | R,E |
| 01041 | 00833 | R,E |
| 01041 | 00836 | R |
| 01041 | 00838 | R |
| 01041 | 00850 | R,E |
| 01041 | 00852 | R,E |
| 01041 | 00855 | R,E |
| 01041 | 00857 | R,E |
| 01041 | 00860 | R |
| 01041 | 00861 | R |
| 01041 | 00863 | R |
| 01041 | 00869 | R |
| 01041 | 00870 | R,E |
| 01041 | 00871 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01041 | 00874 | R |
| 01041 | 00875 | R |
| 01041 | 00880 | R |
| 01041 | 00895 | E |
| 01041 | 00896 | E |
| 01041 | 00897 | E |
| 01041 | 00903 | R |
| 01041 | 00912 | R |
| 01041 | 00916 | R |
| 01041 | 01025 | R,E |
| 01041 | 01026 | R,E |
| 01041 | 01027 | R,E |
| 01041 | 01040 | R,E |
| 01041 | 01042 | R |
| 01041 | 01043 | R,E |
| 01041 | 01088 | R |
| 01041 | 01252 | E |
| 01042 | 00037 | R |
| 01042 | 00273 | R |
| 01042 | 00277 | R |
| 01042 | 00278 | R |
| 01042 | 00280 | R |
| 01042 | 00284 | R |
| 01042 | 00285 | R |
| 01042 | 00290 | R |
| 01042 | 00297 | R |
| 01042 | 00367 | E |
| 01042 | 00423 | R |
| 01042 | 00437 | R |
| 01042 | 00500 | R,E |
| 01042 | 00813 | R |
| 01042 | 00819 | R |
| 01042 | 00833 | R |
| 01042 | 00836 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01042 | 00838 | R |
| 01042 | 00850 | R |
| 01042 | 00852 | R |
| 01042 | 00855 | R |
| 01042 | 00857 | R |
| 01042 | 00860 | R |
| 01042 | 00861 | R |
| 01042 | 00863 | R |
| 01042 | 00869 | R |
| 01042 | 00870 | R |
| 01042 | 00871 | R |
| 01042 | 00874 | R |
| 01042 | 00875 | R |
| 01042 | 00880 | R |
| 01042 | 00897 | R |
| 01042 | 00903 | R |
| 01042 | 00912 | R |
| 01042 | 00916 | R |
| 01042 | 01025 | R |
| 01042 | 01026 | R |
| 01042 | 01027 | R |
| 01042 | 01040 | R |
| 01042 | 01041 | R |
| 01042 | 01043 | R |
| 01042 | 01088 | R |
| 01043 | 00037 | R,E |
| 01043 | 00273 | R,E |
| 01043 | 00277 | R,E |
| 01043 | 00278 | R,E |
| 01043 | 00280 | R,E |
| 01043 | 00284 | R,E |
| 01043 | 00285 | R,E |
| 01043 | 00290 | R,E |
| 01043 | 00297 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 01043 | 00367 | E |
| 01043 | 00423 | R |
| 01043 | 00437 | R,E |
| 01043 | 00500 | R,E |
| 01043 | 00813 | R |
| 01043 | 00819 | R |
| 01043 | 00833 | R,E |
| 01043 | 00836 | R |
| 01043 | 00838 | R |
| 01043 | 00850 | R,E |
| 01043 | 00852 | R,E |
| 01043 | 00855 | R,E |
| 01043 | 00857 | R,E |
| 01043 | 00860 | R |
| 01043 | 00861 | R |
| 01043 | 00863 | R |
| 01043 | 00869 | R |
| 01043 | 00870 | R,E |
| 01043 | 00871 | R,E |
| 01043 | 00874 | R |
| 01043 | 00875 | R |
| 01043 | 00880 | R |
| 01043 | 00897 | R |
| 01043 | 00903 | R |
| 01043 | 00912 | R |
| 01043 | 00916 | R |
| 01043 | 00923 | R,E |
| 01043 | 01025 | R,E |
| 01043 | 01026 | R,E |
| 01043 | 01027 | R,E |
| 01043 | 01040 | R,E |
| 01043 | 01041 | R,E |
| 01043 | 01042 | R |
| 01043 | 01088 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01043 | 01114 | E |
| 01046 | 00367 | E |
| 01046 | 00420 | C,L |
| 01046 | 00425 | C,L |
| 01046 | 00500 | E,L |
| 01046 | 00720 | C |
| 01046 | 00864 | C |
| 01046 | 01089 | C |
| 01046 | 01127 | R |
| 01046 | 01256 | E |
| 01046 | 05352 | E |
| 01047 | 00037 | R |
| 01047 | 00273 | R |
| 01047 | 00274 | R |
| 01047 | 00275 | R |
| 01047 | 00277 | R |
| 01047 | 00278 | R |
| 01047 | 00280 | R |
| 01047 | 00281 | R |
| 01047 | 00282 | R |
| 01047 | 00284 | R |
| 01047 | 00285 | R |
| 01047 | 00290 | R,E |
| 01047 | 00297 | R |
| 01047 | 00367 | E |
| 01047 | 00437 | R,E,M,P |
| 01047 | 00500 | R |
| 01047 | 00819 | R,L |
| 01047 | 00833 | R,E |
| 01047 | 00836 | R,E |
| 01047 | 00850 | R,C,M |
| 01047 | 00852 | R,M |
| 01047 | 00858 | R,E,M |
| 01047 | 00870 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01047 | 00871 | R |
| 01047 | 00875 | R |
| 01047 | 00912 | R,L |
| 01047 | 00923 | R,E,L |
| 01047 | 00924 | R,E |
| 01047 | 01026 | R |
| 01047 | 01027 | R |
| 01047 | 01140 | R,E |
| 01047 | 01141 | R,E |
| 01047 | 01142 | R,E |
| 01047 | 01143 | R,E |
| 01047 | 01144 | R,E |
| 01047 | 01145 | R,E |
| 01047 | 01146 | R,E |
| 01047 | 01147 | R,E |
| 01047 | 01148 | R,E |
| 01047 | 01149 | R,E |
| 01047 | 01252 | R,E,L |
| 01047 | 01254 | R,E,L |
| 01047 | 05348 | R,L |
| 01051 | 00037 | R |
| 01051 | 00273 | R |
| 01051 | 00277 | R |
| 01051 | 00278 | R |
| 01051 | 00280 | R |
| 01051 | 00284 | R |
| 01051 | 00285 | R |
| 01051 | 00297 | R |
| 01051 | 00367 | E |
| 01051 | 00437 | R |
| 01051 | 00500 | R |
| 01051 | 00819 | R |
| 01051 | 00850 | R |
| 01051 | 00858 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01051 | 00863 | R |
| 01051 | 00871 | R |
| 01051 | 00923 | R,E |
| 01051 | 00924 | R,E |
| 01051 | 01025 | R |
| 01051 | 01097 | R |
| 01051 | 01140 | R,E |
| 01051 | 01141 | R,E |
| 01051 | 01142 | R,E |
| 01051 | 01143 | R,E |
| 01051 | 01144 | R,E |
| 01051 | 01145 | R,E |
| 01051 | 01146 | R,E |
| 01051 | 01147 | R,E |
| 01051 | 01148 | R,E |
| 01051 | 01149 | R,E |
| 01051 | 01252 | R |
| 01051 | 01275 | R |
| 01051 | 05348 | R,E |
| 01088 | 00037 | R,L |
| 01088 | 00273 | R,L |
| 01088 | 00277 | R,L |
| 01088 | 00278 | R,L |
| 01088 | 00280 | R,L |
| 01088 | 00284 | R,L |
| 01088 | 00285 | R,L |
| 01088 | 00290 | R,L |
| 01088 | 00297 | R,L |
| 01088 | 00367 | E |
| 01088 | 00500 | R,E,L |
| 01088 | 00819 | R |
| 01088 | 00833 | R,E,L |
| 01088 | 00836 | R,L |
| 01088 | 00850 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01088 | 00852 | R |
| 01088 | 00855 | R |
| 01088 | 00857 | R |
| 01088 | 00870 | R,L |
| 01088 | 00871 | R,L |
| 01088 | 00875 | R,L |
| 01088 | 00891 | E |
| 01088 | 01025 | R,L |
| 01088 | 01026 | R,L |
| 01088 | 01027 | R,L |
| 01088 | 01040 | R,E |
| 01088 | 01041 | R |
| 01088 | 01042 | R |
| 01088 | 01043 | R |
| 01088 | 01126 | E |
| 01089 | 00037 | R,E,L |
| 01089 | 00367 | E |
| 01089 | 00420 | C,L |
| 01089 | 00425 | E,L |
| 01089 | 00500 | R,E,L |
| 01089 | 00819 | R |
| 01089 | 00850 | R |
| 01089 | 00864 | E,C |
| 01089 | 01046 | C |
| 01089 | 01127 | C |
| 01089 | 01148 | E,L |
| 01089 | 01256 | E |
| 01089 | 05352 | E |
| 01089 | 09238 | C |
| 01097 | 00037 | R,E |
| 01097 | 00367 | E |
| 01097 | 00437 | R,E |
| 01097 | 00500 | R,E |
| 01097 | 00737 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01097 | 00775 | R,E |
| 01097 | 00819 | R |
| 01097 | 00850 | R |
| 01097 | 00852 | R,E |
| 01097 | 00857 | R,E |
| 01097 | 00860 | R,E |
| 01097 | 00861 | R,E |
| 01097 | 00862 | R,E |
| 01097 | 00863 | R,E |
| 01097 | 00864 | R,E |
| 01097 | 00865 | R,E |
| 01097 | 01051 | R,E |
| 01097 | 01098 | R,E |
| 01097 | 01112 | R |
| 01097 | 01122 | R |
| 01097 | 01252 | R |
| 01098 | 00259 | R |
| 01098 | 00367 | E |
| 01098 | 00420 | R |
| 01098 | 00437 | R |
| 01098 | 00819 | R |
| 01098 | 00850 | R |
| 01098 | 01097 | R |
| 01098 | 01252 | R |
| 01100 | 00037 | R |
| 01100 | 00273 | R |
| 01100 | 00277 | R |
| 01100 | 00278 | R |
| 01100 | 00280 | R |
| 01100 | 00284 | R |
| 01100 | 00285 | R |
| 01100 | 00297 | R |
| 01100 | 00367 | E |
| 01100 | 00500 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01100 | 00850 | R |
| 01101 | 00367 | E |
| 01101 | 00500 | E |
| 01102 | 00367 | E |
| 01102 | 00500 | E |
| 01103 | 00367 | E |
| 01103 | 00500 | E |
| 01104 | 00367 | E |
| 01104 | 00500 | E |
| 01105 | 00367 | E |
| 01105 | 00500 | E |
| 01106 | 00367 | E |
| 01106 | 00500 | E |
| 01107 | 00367 | E |
| 01107 | 00500 | E |
| 01112 | 00037 | R,E |
| 01112 | 00256 | R |
| 01112 | 00273 | R |
| 01112 | 00277 | R |
| 01112 | 00278 | R |
| 01112 | 00280 | R |
| 01112 | 00284 | R |
| 01112 | 00285 | R |
| 01112 | 00290 | R |
| 01112 | 00297 | R |
| 01112 | 00367 | E |
| 01112 | 00420 | R |
| 01112 | 00423 | R |
| 01112 | 00424 | R |
| 01112 | 00500 | R,E |
| 01112 | 00775 | R |
| 01112 | 00819 | R,E,L |
| 01112 | 00833 | R |
| 01112 | 00836 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01112 | 00838 | R |
| 01112 | 00850 | R,M |
| 01112 | 00870 | R |
| 01112 | 00871 | R |
| 01112 | 00875 | R |
| 01112 | 00880 | R |
| 01112 | 00905 | R |
| 01112 | 00914 | R,E,L |
| 01112 | 00921 | R,E,L |
| 01112 | 00922 | R,L |
| 01112 | 01025 | R |
| 01112 | 01026 | R |
| 01112 | 01027 | R |
| 01112 | 01097 | R |
| 01112 | 01122 | R |
| 01112 | 01252 | R,E,L |
| 01112 | 01257 | R,E |
| 01112 | 05353 | R,E |
| 01114 | 00037 | E |
| 01114 | 00367 | E |
| 01114 | 00437 | E |
| 01114 | 00500 | E |
| 01114 | 00819 | R,E |
| 01114 | 00836 | E |
| 01114 | 00850 | R,E |
| 01114 | 00904 | E |
| 01114 | 01043 | E |
| 01114 | 01115 | E |
| 01114 | 28709 | E |
| 01115 | 00037 | E,L |
| 01115 | 00367 | E |
| 01115 | 00437 | E |
| 01115 | 00500 | E,L |
| 01115 | 00836 | E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01115 | 00903 | E |
| 01115 | 01114 | E |
| 01122 | 00037 | R |
| 01122 | 00256 | R |
| 01122 | 00273 | R |
| 01122 | 00277 | R |
| 01122 | 00278 | R |
| 01122 | 00280 | R |
| 01122 | 00284 | R |
| 01122 | 00285 | R |
| 01122 | 00290 | R |
| 01122 | 00297 | R |
| 01122 | 00367 | E |
| 01122 | 00420 | R |
| 01122 | 00423 | R |
| 01122 | 00424 | R |
| 01122 | 00500 | R |
| 01122 | 00775 | R |
| 01122 | 00819 | R,E,L |
| 01122 | 00833 | R |
| 01122 | 00836 | R |
| 01122 | 00838 | R |
| 01122 | 00850 | R,M |
| 01122 | 00870 | R |
| 01122 | 00871 | R |
| 01122 | 00875 | R |
| 01122 | 00880 | R |
| 01122 | 00905 | R |
| 01122 | 00914 | R,E,L |
| 01122 | 00921 | R,L |
| 01122 | 00922 | R,E,L |
| 01122 | 01025 | R |
| 01122 | 01026 | R |
| 01122 | 01027 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01122 | 01097 | R |
| 01122 | 01112 | R |
| 01122 | 01252 | R,E,L |
| 01122 | 01257 | R,E |
| 01122 | 05353 | R,E |
| 01123 | 00037 | R,E |
| 01123 | 00367 | E |
| 01123 | 00500 | R,E |
| 01123 | 00819 | R,E,L |
| 01123 | 00848 | R,E,M |
| 01123 | 01124 | R,E,L |
| 01123 | 01125 | R,E,L |
| 01123 | 01148 | R,E |
| 01123 | 01168 | R,E |
| 01123 | 01251 | R,E,L |
| 01123 | 01252 | R,E,L |
| 01123 | 01283 | R |
| 01123 | 05347 | R,E,L |
| 01124 | 00037 | R,E,L |
| 01124 | 00367 | E |
| 01124 | 00500 | R,E,L |
| 01124 | 01123 | R,E,L |
| 01124 | 01125 | R,E |
| 01124 | 01158 | E,L |
| 01124 | 01168 | R,E |
| 01124 | 01251 | R,E |
| 01124 | 01283 | R |
| 01124 | 05347 | R,E |
| 01125 | 00367 | E |
| 01125 | 00500 | R,E,L |
| 01125 | 01123 | R,E,L |
| 01125 | 01124 | R,E |
| 01125 | 01251 | R,E |
| 01125 | 01283 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01125 | 05347 | R,E |
| 01126 | 00037 | E,L |
| 01126 | 00367 | E |
| 01126 | 00437 | E |
| 01126 | 00500 | E,L |
| 01126 | 00819 | E |
| 01126 | 00833 | E,L |
| 01126 | 00850 | E |
| 01126 | 01088 | E |
| 01126 | 01252 | E |
| 01126 | 13121 | E,L |
| 01127 | 00420 | R |
| 01127 | 00864 | R |
| 01127 | 01046 | R |
| 01127 | 01089 | C |
| 01127 | 01256 | C |
| 01129 | 00500 | R,E |
| 01129 | 01130 | R,E |
| 01129 | 01258 | R,E |
| 01129 | 05354 | R,E |
| 01130 | 00037 | R |
| 01130 | 00500 | R,E |
| 01130 | 00819 | R,E |
| 01130 | 00850 | R,E |
| 01130 | 01129 | R,E |
| 01130 | 01163 | R,E |
| 01130 | 01252 | R,E |
| 01130 | 01258 | R,E |
| 01130 | 05354 | R,E |
| 01131 | 00037 | R,E |
| 01131 | 00367 | E |
| 01131 | 00500 | R |
| 01131 | 00878 | R,E |
| 01131 | 00915 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01131 | 01025 | R |
| 01131 | 01251 | R |
| 01131 | 01283 | R |
| 01131 | 05347 | R,E |
| 01132 | 00037 | R |
| 01132 | 00500 | R,E |
| 01132 | 00819 | R,E |
| 01132 | 00850 | R,E |
| 01132 | 01133 | R,E |
| 01132 | 01252 | R,E |
| 01133 | 00500 | R,E |
| 01133 | 01132 | R,E |
| 01137 | 00037 | E |
| 01137 | 00500 | E |
| 01137 | 00806 | E |
| 01137 | 00819 | E |
| 01137 | 01252 | R,E |
| 01140 | 00037 | E |
| 01140 | 00273 | R,E |
| 01140 | 00277 | R,E |
| 01140 | 00278 | R,E |
| 01140 | 00280 | R,E |
| 01140 | 00284 | R,E |
| 01140 | 00285 | R,E |
| 01140 | 00297 | R,E |
| 01140 | 00367 | E |
| 01140 | 00425 | R,E |
| 01140 | 00437 | R,E,M |
| 01140 | 00500 | R,E |
| 01140 | 00808 | R,E,M |
| 01140 | 00819 | R,E,L |
| 01140 | 00850 | R,E,M,P |
| 01140 | 00858 | R,E,M |
| 01140 | 00860 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 01140 | 00863 | R,E |
| 01140 | 00871 | R,E |
| 01140 | 00872 | R,E,M |
| 01140 | 00901 | R,E,L |
| 01140 | 00902 | R,E,L |
| 01140 | 00923 | R,E,L |
| 01140 | 00924 | R,E |
| 01140 | 01013 | E |
| 01140 | 01047 | R,E |
| 01140 | 01051 | R,E |
| 01140 | 01141 | R |
| 01140 | 01142 | R |
| 01140 | 01143 | R |
| 01140 | 01144 | R |
| 01140 | 01145 | R |
| 01140 | 01146 | R |
| 01140 | 01147 | R |
| 01140 | 01148 | R |
| 01140 | 01149 | R |
| 01140 | 01153 | R,E |
| 01140 | 01154 | R,E |
| 01140 | 01155 | R,E |
| 01140 | 01156 | R,E |
| 01140 | 01157 | R,E |
| 01140 | 01160 | R,E |
| 01140 | 01161 | R,E,L |
| 01140 | 01162 | R,E |
| 01140 | 01164 | R,E |
| 01140 | 01252 | R,E,L |
| 01140 | 01275 | R,E |
| 01140 | 04909 | R,E,L |
| 01140 | 04971 | R,E |
| 01140 | 05123 | E |
| 01140 | 05348 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01140 | 09044 | R,E,L |
| 01140 | 09049 | R,E |
| 01140 | 09061 | R,E |
| 01140 | 16804 | R,E |
| 01140 | 17248 | R,E,L |
| 01141 | 00037 | R,E |
| 01141 | 00273 | E |
| 01141 | 00277 | R,E |
| 01141 | 00278 | R,E |
| 01141 | 00280 | R,E |
| 01141 | 00284 | R,E |
| 01141 | 00285 | R,E |
| 01141 | 00297 | R,E |
| 01141 | 00367 | E |
| 01141 | 00437 | R,E,M |
| 01141 | 00500 | R,E |
| 01141 | 00819 | R,E,L |
| 01141 | 00850 | R,E,M,P |
| 01141 | 00858 | R,E,M |
| 01141 | 00863 | R,E |
| 01141 | 00871 | R,E |
| 01141 | 00872 | R,E,M |
| 01141 | 00923 | R,E,L |
| 01141 | 00924 | R,E |
| 01141 | 01047 | R,E |
| 01141 | 01051 | R,E |
| 01141 | 01140 | R |
| 01141 | 01142 | R |
| 01141 | 01143 | R |
| 01141 | 01144 | R |
| 01141 | 01145 | R |
| 01141 | 01146 | R |
| 01141 | 01147 | R |
| 01141 | 01148 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01141 | 01149 | R |
| 01141 | 01153 | R,E |
| 01141 | 01154 | R,E |
| 01141 | 01155 | R,E |
| 01141 | 01156 | R,E |
| 01141 | 01157 | R,E |
| 01141 | 01160 | R,E |
| 01141 | 01161 | R,E,L |
| 01141 | 01162 | R,E |
| 01141 | 01252 | R,E,L |
| 01141 | 01275 | R,E |
| 01141 | 04909 | R,E,L |
| 01141 | 04971 | R,E |
| 01141 | 05123 | E |
| 01141 | 05348 | R,E,L |
| 01141 | 09044 | R,E,L |
| 01141 | 09049 | R,E |
| 01141 | 09061 | R,E |
| 01141 | 17248 | R,E,L |
| 01142 | 00037 | R,E |
| 01142 | 00273 | R,E |
| 01142 | 00277 | E |
| 01142 | 00278 | R,E |
| 01142 | 00280 | R,E |
| 01142 | 00284 | R,E |
| 01142 | 00285 | R,E |
| 01142 | 00297 | R,E |
| 01142 | 00367 | E |
| 01142 | 00437 | R,E,M |
| 01142 | 00500 | R,E |
| 01142 | 00819 | R,E,L |
| 01142 | 00850 | R,E,M,P |
| 01142 | 00858 | R,E,M |
| 01142 | 00863 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01142 | 00865 | R,E |
| 01142 | 00871 | R,E |
| 01142 | 00872 | R,E,M |
| 01142 | 00923 | R,E,L |
| 01142 | 00924 | R,E |
| 01142 | 01047 | R,E |
| 01142 | 01051 | R,E |
| 01142 | 01140 | R |
| 01142 | 01141 | R |
| 01142 | 01143 | R |
| 01142 | 01144 | R |
| 01142 | 01145 | R |
| 01142 | 01146 | R |
| 01142 | 01147 | R |
| 01142 | 01148 | R |
| 01142 | 01149 | R |
| 01142 | 01153 | R,E |
| 01142 | 01154 | R,E |
| 01142 | 01155 | R,E |
| 01142 | 01156 | R,E |
| 01142 | 01157 | R,E |
| 01142 | 01160 | R,E |
| 01142 | 01161 | R,E,L |
| 01142 | 01162 | R,E |
| 01142 | 01252 | R,E,L |
| 01142 | 01275 | R,E |
| 01142 | 04909 | R,E,L |
| 01142 | 04971 | R,E |
| 01142 | 05123 | E |
| 01142 | 05348 | R,E,L |
| 01142 | 09044 | R,E,L |
| 01142 | 09049 | R,E |
| 01142 | 09061 | R,E |
| 01142 | 17248 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01143 | 00037 | R,E |
| 01143 | 00273 | R,E |
| 01143 | 00277 | R,E |
| 01143 | 00278 | E |
| 01143 | 00280 | R,E |
| 01143 | 00284 | R,E |
| 01143 | 00285 | R,E |
| 01143 | 00297 | R,E |
| 01143 | 00367 | E |
| 01143 | 00437 | R,E,M |
| 01143 | 00500 | R,E |
| 01143 | 00819 | R,E,L |
| 01143 | 00850 | R,E,M,P |
| 01143 | 00858 | R,E,M |
| 01143 | 00863 | R,E |
| 01143 | 00865 | R,E |
| 01143 | 00871 | R,E |
| 01143 | 00872 | R,E,M |
| 01143 | 00923 | R,E,L |
| 01143 | 00924 | R,E |
| 01143 | 01047 | R,E |
| 01143 | 01051 | R,E |
| 01143 | 01140 | R |
| 01143 | 01141 | R |
| 01143 | 01142 | R |
| 01143 | 01144 | R |
| 01143 | 01145 | R |
| 01143 | 01146 | R |
| 01143 | 01147 | R |
| 01143 | 01148 | R |
| 01143 | 01149 | R |
| 01143 | 01153 | R,E |
| 01143 | 01154 | R,E |
| 01143 | 01155 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01143 | 01156 | R,E |
| 01143 | 01157 | R,E |
| 01143 | 01160 | R,E |
| 01143 | 01161 | R,E,L |
| 01143 | 01162 | R,E |
| 01143 | 01252 | R,E,L |
| 01143 | 01275 | R,E |
| 01143 | 04909 | R,E,L |
| 01143 | 04971 | R,E |
| 01143 | 05123 | E |
| 01143 | 05348 | R,E,L |
| 01143 | 09044 | R,E,L |
| 01143 | 09049 | R,E |
| 01143 | 09061 | R,E |
| 01143 | 17248 | R,E,L |
| 01144 | 00037 | R,E |
| 01144 | 00273 | R,E |
| 01144 | 00277 | R,E |
| 01144 | 00278 | R,E |
| 01144 | 00280 | E |
| 01144 | 00284 | R,E |
| 01144 | 00285 | R,E |
| 01144 | 00297 | R,E |
| 01144 | 00367 | E |
| 01144 | 00437 | R,E,M |
| 01144 | 00500 | R,E |
| 01144 | 00819 | R,E,L |
| 01144 | 00850 | R,E,M,P |
| 01144 | 00858 | R,E,M |
| 01144 | 00863 | R,E |
| 01144 | 00871 | R,E |
| 01144 | 00872 | R,E,M |
| 01144 | 00923 | R,E,L |
| 01144 | 00924 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01144 | 01047 | R,E |
| 01144 | 01051 | R,E |
| 01144 | 01140 | R |
| 01144 | 01141 | R |
| 01144 | 01142 | R |
| 01144 | 01143 | R |
| 01144 | 01145 | R |
| 01144 | 01146 | R |
| 01144 | 01147 | R |
| 01144 | 01148 | R |
| 01144 | 01149 | R |
| 01144 | 01153 | R,E |
| 01144 | 01154 | R,E |
| 01144 | 01155 | R,E |
| 01144 | 01156 | R,E |
| 01144 | 01157 | R,E |
| 01144 | 01160 | R,E |
| 01144 | 01161 | R,E,L |
| 01144 | 01162 | R,E |
| 01144 | 01252 | R,E,L |
| 01144 | 01275 | R,E |
| 01144 | 04909 | R,E,L |
| 01144 | 04971 | R,E |
| 01144 | 05123 | E |
| 01144 | 05348 | R,E,L |
| 01144 | 09044 | R,E,L |
| 01144 | 09049 | R,E |
| 01144 | 09061 | R,E |
| 01144 | 17248 | R,E,L |
| 01145 | 00037 | R,E |
| 01145 | 00273 | R,E |
| 01145 | 00277 | R,E |
| 01145 | 00278 | R,E |
| 01145 | 00280 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01145 | 00284 | E |
| 01145 | 00285 | R,E |
| 01145 | 00297 | R,E |
| 01145 | 00367 | E |
| 01145 | 00437 | R,E,M |
| 01145 | 00500 | R,E |
| 01145 | 00819 | R,E,L |
| 01145 | 00850 | R,E,M,P |
| 01145 | 00858 | R,E,M |
| 01145 | 00860 | R,E |
| 01145 | 00863 | R,E |
| 01145 | 00871 | R,E |
| 01145 | 00872 | R,E,M |
| 01145 | 00923 | R,E,L |
| 01145 | 00924 | R,E |
| 01145 | 01047 | R,E |
| 01145 | 01051 | R,E |
| 01145 | 01140 | R |
| 01145 | 01141 | R |
| 01145 | 01142 | R |
| 01145 | 01143 | R |
| 01145 | 01144 | R |
| 01145 | 01146 | R |
| 01145 | 01147 | R |
| 01145 | 01148 | R |
| 01145 | 01149 | R |
| 01145 | 01153 | R,E |
| 01145 | 01154 | R,E |
| 01145 | 01155 | R,E |
| 01145 | 01156 | R,E |
| 01145 | 01157 | R,E |
| 01145 | 01160 | R,E |
| 01145 | 01161 | R,E,L |
| 01145 | 01162 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01145 | 01252 | R,E,L |
| 01145 | 01275 | R,E |
| 01145 | 04909 | R,E,L |
| 01145 | 04971 | R,E |
| 01145 | 05123 | E |
| 01145 | 05348 | R,E,L |
| 01145 | 09044 | R,E,L |
| 01145 | 09049 | R,E |
| 01145 | 09061 | R,E |
| 01145 | 17248 | R,E,L |
| 01146 | 00037 | R,E |
| 01146 | 00273 | R,E |
| 01146 | 00277 | R,E |
| 01146 | 00278 | R,E |
| 01146 | 00280 | R,E |
| 01146 | 00284 | R,E |
| 01146 | 00285 | E |
| 01146 | 00297 | R,E |
| 01146 | 00367 | E |
| 01146 | 00437 | R,E,M |
| 01146 | 00500 | R,E |
| 01146 | 00819 | R,E,L |
| 01146 | 00850 | R,E,M,P |
| 01146 | 00858 | R,E,M |
| 01146 | 00860 | R,E |
| 01146 | 00863 | R,E |
| 01146 | 00871 | R,E |
| 01146 | 00872 | R,E,M |
| 01146 | 00923 | R,E,L |
| 01146 | 00924 | R,E |
| 01146 | 01047 | R,E |
| 01146 | 01051 | R,E |
| 01146 | 01140 | R |
| 01146 | 01141 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01146 | 01142 | R |
| 01146 | 01143 | R |
| 01146 | 01144 | R |
| 01146 | 01145 | R |
| 01146 | 01147 | R |
| 01146 | 01148 | R |
| 01146 | 01149 | R |
| 01146 | 01153 | R,E |
| 01146 | 01154 | R,E |
| 01146 | 01155 | R,E |
| 01146 | 01156 | R,E |
| 01146 | 01157 | R,E |
| 01146 | 01160 | R,E |
| 01146 | 01161 | R,E,L |
| 01146 | 01162 | R,E |
| 01146 | 01252 | R,E,L |
| 01146 | 01275 | R,E |
| 01146 | 04909 | R,E,L |
| 01146 | 04971 | R,E |
| 01146 | 05123 | E |
| 01146 | 05348 | R,E,L |
| 01146 | 09044 | R,E,L |
| 01146 | 09049 | R,E |
| 01146 | 09061 | R,E |
| 01146 | 17248 | R,E,L |
| 01147 | 00037 | R,E |
| 01147 | 00273 | R,E |
| 01147 | 00277 | R,E |
| 01147 | 00278 | R,E |
| 01147 | 00280 | R,E |
| 01147 | 00284 | R,E |
| 01147 | 00285 | R,E |
| 01147 | 00297 | E |
| 01147 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 01147 | 00437 | R,E,M |
| 01147 | 00500 | R,E |
| 01147 | 00819 | R,E,L |
| 01147 | 00850 | R,E,M,P |
| 01147 | 00858 | R,E,M |
| 01147 | 00863 | R,E |
| 01147 | 00870 | R,E |
| 01147 | 00871 | R,E |
| 01147 | 00872 | R,E,M |
| 01147 | 00923 | R,E,L |
| 01147 | 00924 | R,E |
| 01147 | 01047 | R,E |
| 01147 | 01051 | R,E |
| 01147 | 01140 | R |
| 01147 | 01141 | R |
| 01147 | 01142 | R |
| 01147 | 01143 | R |
| 01147 | 01144 | R |
| 01147 | 01145 | R |
| 01147 | 01146 | R |
| 01147 | 01148 | R |
| 01147 | 01149 | R |
| 01147 | 01153 | R,E |
| 01147 | 01154 | R,E |
| 01147 | 01155 | R,E |
| 01147 | 01156 | R,E |
| 01147 | 01157 | R,E |
| 01147 | 01160 | R,E |
| 01147 | 01161 | R,E,L |
| 01147 | 01162 | R,E |
| 01147 | 01252 | R,E,L |
| 01147 | 01275 | R,E |
| 01147 | 04909 | R,E,L |
| 01147 | 04971 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01147 | 05123 | E |
| 01147 | 05348 | R,E,L |
| 01147 | 09044 | R,E,L |
| 01147 | 09049 | R,E |
| 01147 | 09061 | R,E |
| 01147 | 17248 | R,E,L |
| 01148 | 00037 | R,E |
| 01148 | 00273 | R,E |
| 01148 | 00274 | R,E |
| 01148 | 00275 | R,E |
| 01148 | 00277 | R,E |
| 01148 | 00278 | R,E |
| 01148 | 00280 | R,E |
| 01148 | 00281 | R,E |
| 01148 | 00282 | R,E |
| 01148 | 00284 | R,E |
| 01148 | 00285 | R,E |
| 01148 | 00290 | R,E |
| 01148 | 00297 | R,E |
| 01148 | 00367 | E |
| 01148 | 00425 | R,E |
| 01148 | 00437 | R,E,M |
| 01148 | 00500 | E |
| 01148 | 00808 | R,E,M |
| 01148 | 00819 | R,E,L |
| 01148 | 00848 | R,E,M |
| 01148 | 00849 | R,E |
| 01148 | 00850 | R,E,M,P |
| 01148 | 00858 | R,E,M |
| 01148 | 00860 | R,E |
| 01148 | 00861 | R,E,M |
| 01148 | 00863 | R,E |
| 01148 | 00865 | R,E |
| 01148 | 00867 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01148 | 00871 | R,E |
| 01148 | 00872 | R,E,M |
| 01148 | 00901 | R,E,L |
| 01148 | 00902 | R,E,L |
| 01148 | 00912 | E,L |
| 01148 | 00916 | E,L |
| 01148 | 00920 | E,L |
| 01148 | 00923 | R,E,L |
| 01148 | 00924 | R,E |
| 01148 | 01027 | R,E |
| 01148 | 01047 | R,E |
| 01148 | 01051 | R,E |
| 01148 | 01089 | E,L |
| 01148 | 01123 | R,E |
| 01148 | 01140 | R |
| 01148 | 01141 | R |
| 01148 | 01142 | R |
| 01148 | 01143 | R |
| 01148 | 01144 | R |
| 01148 | 01145 | R |
| 01148 | 01146 | R |
| 01148 | 01147 | R |
| 01148 | 01149 | R |
| 01148 | 01153 | R,E |
| 01148 | 01154 | R,E |
| 01148 | 01155 | R,E |
| 01148 | 01156 | R,E |
| 01148 | 01157 | R,E |
| 01148 | 01158 | R,E |
| 01148 | 01159 | R,E |
| 01148 | 01160 | R,E |
| 01148 | 01161 | R,E,L |
| 01148 | 01162 | R,E |
| 01148 | 01163 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01148 | 01164 | R,E |
| 01148 | 01252 | R,E,L |
| 01148 | 01275 | R,E |
| 01148 | 01281 | R,E |
| 01148 | 04899 | R,E |
| 01148 | 04909 | R,E,L |
| 01148 | 04971 | R,E |
| 01148 | 05123 | E |
| 01148 | 05210 | R,E |
| 01148 | 05346 | R,E,L |
| 01148 | 05347 | R,E,L |
| 01148 | 05348 | R,E,L |
| 01148 | 05349 | R,E,L |
| 01148 | 05350 | R,E,L |
| 01148 | 05351 | R,E,L |
| 01148 | 05352 | R,E,L |
| 01148 | 05353 | R,E |
| 01148 | 05354 | R,E |
| 01148 | 08482 | R,E |
| 01148 | 09044 | R,E,L |
| 01148 | 09048 | R,E |
| 01148 | 09049 | R,E |
| 01148 | 09061 | R,E |
| 01148 | 09238 | R,E,L |
| 01148 | 12712 | R,E |
| 01148 | 16804 | R,E |
| 01148 | 17248 | R,E,L |
| 01149 | 00037 | R,E |
| 01149 | 00273 | R,E |
| 01149 | 00277 | R,E |
| 01149 | 00278 | R,E |
| 01149 | 00280 | R,E |
| 01149 | 00284 | R,E |
| 01149 | 00285 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 01149 | 00297 | R,E |
| 01149 | 00367 | E |
| 01149 | 00437 | R,E,M |
| 01149 | 00500 | R,E |
| 01149 | 00819 | R,E,L |
| 01149 | 00850 | R,E,M |
| 01149 | 00858 | R,E,M |
| 01149 | 00861 | R,E,M |
| 01149 | 00863 | R,E |
| 01149 | 00871 | E |
| 01149 | 00872 | R,E,M |
| 01149 | 00923 | R,E,L |
| 01149 | 00924 | R,E |
| 01149 | 01047 | R,E |
| 01149 | 01051 | R,E |
| 01149 | 01140 | R |
| 01149 | 01141 | R |
| 01149 | 01142 | R |
| 01149 | 01143 | R |
| 01149 | 01144 | R |
| 01149 | 01145 | R |
| 01149 | 01146 | R |
| 01149 | 01147 | R |
| 01149 | 01148 | R |
| 01149 | 01153 | R,E |
| 01149 | 01154 | R,E |
| 01149 | 01155 | R,E |
| 01149 | 01156 | R,E |
| 01149 | 01157 | R,E |
| 01149 | 01160 | R,E |
| 01149 | 01161 | R,E,L |
| 01149 | 01162 | R,E |
| 01149 | 01252 | R,E,L |
| 01149 | 01275 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01149 | 04909 | R,E,L |
| 01149 | 04971 | R,E |
| 01149 | 05123 | E |
| 01149 | 05348 | R,E,L |
| 01149 | 09044 | R,E,L |
| 01149 | 09049 | R,E |
| 01149 | 09061 | R,E |
| 01149 | 17248 | R,E,L |
| 01153 | 00808 | R,E,M |
| 01153 | 00819 | R,E,L |
| 01153 | 00850 | R,E,M |
| 01153 | 00852 | R,E,M |
| 01153 | 00858 | R,E,M |
| 01153 | 00859 | R,E,M |
| 01153 | 00867 | R,E,M |
| 01153 | 00872 | R,E,M |
| 01153 | 00912 | R,E,L |
| 01153 | 00923 | R,E,L |
| 01153 | 00924 | R,E |
| 01153 | 01140 | R,E |
| 01153 | 01141 | R,E |
| 01153 | 01142 | R,E |
| 01153 | 01143 | R,E |
| 01153 | 01144 | R,E |
| 01153 | 01145 | R,E |
| 01153 | 01146 | R,E |
| 01153 | 01147 | R,E |
| 01153 | 01148 | R,E |
| 01153 | 01149 | R,E |
| 01153 | 01154 | R,E |
| 01153 | 01155 | R,E |
| 01153 | 01156 | R,E |
| 01153 | 01157 | R,E |
| 01153 | 01160 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01153 | 01161 | R,E,L |
| 01153 | 01162 | R,E |
| 01153 | 01282 | R,E |
| 01153 | 04909 | R,E,L |
| 01153 | 04971 | R,E |
| 01153 | 05346 | R,E,L |
| 01153 | 05348 | R,E,L |
| 01153 | 09044 | R,E,L |
| 01153 | 09049 | R,E |
| 01153 | 09061 | R,E |
| 01153 | 17248 | R,E,L |
| 01154 | 00808 | R,E,M |
| 01154 | 00819 | R,E,L |
| 01154 | 00848 | R,M |
| 01154 | 00849 | R,E |
| 01154 | 00858 | R,E,M |
| 01154 | 00859 | R,E,M |
| 01154 | 00867 | R,E,M |
| 01154 | 00872 | R,E,M |
| 01154 | 00915 | R,E,L |
| 01154 | 00923 | R,E,L |
| 01154 | 00924 | R,E |
| 01154 | 01140 | R,E |
| 01154 | 01141 | R,E |
| 01154 | 01142 | R,E |
| 01154 | 01143 | R,E |
| 01154 | 01144 | R,E |
| 01154 | 01145 | R,E |
| 01154 | 01146 | R,E |
| 01154 | 01147 | R,E |
| 01154 | 01148 | R,E |
| 01154 | 01149 | R,E |
| 01154 | 01153 | R,E |
| 01154 | 01155 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01154 | 01156 | R,E |
| 01154 | 01157 | R,E |
| 01154 | 01160 | R,E |
| 01154 | 01161 | R,E,L |
| 01154 | 01162 | R,E |
| 01154 | 01283 | R,E |
| 01154 | 04909 | R,E,L |
| 01154 | 04971 | R,E |
| 01154 | 05123 | E |
| 01154 | 05347 | R,E,L |
| 01154 | 05348 | R,E,L |
| 01154 | 09044 | R,E,L |
| 01154 | 09049 | R,E |
| 01154 | 09061 | R,E |
| 01154 | 16804 | R,E |
| 01154 | 17248 | R,E,L |
| 01155 | 00819 | R,E,L |
| 01155 | 00858 | R,E,M |
| 01155 | 00859 | R,E,M |
| 01155 | 00867 | R,E,M |
| 01155 | 00872 | R,E,M |
| 01155 | 00920 | R,E,L |
| 01155 | 00923 | R,E,L |
| 01155 | 00924 | R,E |
| 01155 | 01140 | R,E |
| 01155 | 01141 | R,E |
| 01155 | 01142 | R,E |
| 01155 | 01143 | R,E |
| 01155 | 01144 | R,E |
| 01155 | 01145 | R,E |
| 01155 | 01146 | R,E |
| 01155 | 01147 | R,E |
| 01155 | 01148 | R,E |
| 01155 | 01149 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01155 | 01153 | R,E |
| 01155 | 01154 | R,E |
| 01155 | 01156 | R,E |
| 01155 | 01157 | R,E |
| 01155 | 01160 | R,E |
| 01155 | 01161 | R,E,L |
| 01155 | 01162 | R,E |
| 01155 | 01175 | R |
| 01155 | 01281 | R,E |
| 01155 | 04909 | R,E,L |
| 01155 | 04971 | R,E |
| 01155 | 05348 | R,E,L |
| 01155 | 05350 | R,E,L |
| 01155 | 09044 | R,E,L |
| 01155 | 09049 | R,E |
| 01155 | 09061 | R,E |
| 01155 | 16804 | R,E |
| 01155 | 17248 | R,E,L |
| 01156 | 00819 | R,E,L |
| 01156 | 00858 | R,E,M |
| 01156 | 00859 | R,E,M |
| 01156 | 00901 | R,E,L |
| 01156 | 00902 | R,E,L |
| 01156 | 00923 | R,E,L |
| 01156 | 00924 | R,E |
| 01156 | 01140 | R,E |
| 01156 | 01141 | R,E |
| 01156 | 01142 | R,E |
| 01156 | 01143 | R,E |
| 01156 | 01144 | R,E |
| 01156 | 01145 | R,E |
| 01156 | 01146 | R,E |
| 01156 | 01147 | R,E |
| 01156 | 01148 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01156 | 01149 | R,E |
| 01156 | 01153 | R,E |
| 01156 | 01154 | R,E |
| 01156 | 01155 | R,E |
| 01156 | 01157 | R,E |
| 01156 | 01160 | R,E |
| 01156 | 04971 | R,E |
| 01156 | 05123 | E |
| 01156 | 05348 | R,E,L |
| 01156 | 05353 | R,E |
| 01156 | 12712 | R,E |
| 01156 | 16804 | R,E |
| 01157 | 00819 | R,E,L |
| 01157 | 00858 | R,E,M |
| 01157 | 00859 | R,E,M |
| 01157 | 00901 | R,E,L |
| 01157 | 00902 | R,E,L |
| 01157 | 00923 | R,E,L |
| 01157 | 00924 | R,E |
| 01157 | 01140 | R,E |
| 01157 | 01141 | R,E |
| 01157 | 01142 | R,E |
| 01157 | 01143 | R,E |
| 01157 | 01144 | R,E |
| 01157 | 01145 | R,E |
| 01157 | 01146 | R,E |
| 01157 | 01147 | R,E |
| 01157 | 01148 | R,E |
| 01157 | 01149 | R,E |
| 01157 | 01153 | R,E |
| 01157 | 01154 | R,E |
| 01157 | 01155 | R,E |
| 01157 | 01156 | R,E |
| 01157 | 01160 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01157 | 04971 | R,E |
| 01157 | 05123 | E |
| 01157 | 05348 | R,E,L |
| 01157 | 05353 | R,E |
| 01157 | 12712 | R,E |
| 01157 | 16804 | R,E |
| 01158 | 00808 | R,M |
| 01158 | 00819 | R,E,L |
| 01158 | 00848 | R,E,M |
| 01158 | 00849 | R |
| 01158 | 00923 | R,E,L |
| 01158 | 01124 | E,L |
| 01158 | 01148 | R,E |
| 01158 | 05347 | R,E,L |
| 01158 | 05348 | R,E,L |
| 01159 | 01148 | R,E |
| 01159 | 05210 | E |
| 01160 | 00819 | R,E,L |
| 01160 | 00858 | R,E,M |
| 01160 | 00859 | R,E,M |
| 01160 | 00867 | R,E,M |
| 01160 | 00923 | R,E,L |
| 01160 | 00924 | R,E |
| 01160 | 01140 | R,E |
| 01160 | 01141 | R,E |
| 01160 | 01142 | R,E |
| 01160 | 01143 | R,E |
| 01160 | 01144 | R,E |
| 01160 | 01145 | R,E |
| 01160 | 01146 | R,E |
| 01160 | 01147 | R,E |
| 01160 | 01148 | R,E |
| 01160 | 01149 | R,E |
| 01160 | 01153 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01160 | 01154 | R,E |
| 01160 | 01155 | R,E |
| 01160 | 01156 | R,E |
| 01160 | 01157 | R,E |
| 01160 | 01161 | R,E,L |
| 01160 | 01162 | R,E |
| 01160 | 04909 | R,E,L |
| 01160 | 04971 | R,E |
| 01160 | 05123 | E |
| 01160 | 05348 | R,E,L |
| 01160 | 09044 | R,E,L |
| 01160 | 09049 | R,E |
| 01160 | 09061 | R,E |
| 01160 | 17248 | R,E,L |
| 01161 | 00259 | E |
| 01161 | 00838 | R,E,L |
| 01161 | 00858 | R,E |
| 01161 | 00859 | R,E |
| 01161 | 00923 | R,E |
| 01161 | 00924 | R,E,L |
| 01161 | 01140 | R,E,L |
| 01161 | 01141 | R,E,L |
| 01161 | 01142 | R,E,L |
| 01161 | 01143 | R,E,L |
| 01161 | 01144 | R,E,L |
| 01161 | 01145 | R,E,L |
| 01161 | 01146 | R,E,L |
| 01161 | 01147 | R,E,L |
| 01161 | 01148 | R,E,L |
| 01161 | 01149 | R,E,L |
| 01161 | 01153 | R,E,L |
| 01161 | 01154 | R,E,L |
| 01161 | 01155 | R,E,L |
| 01161 | 01160 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01161 | 04909 | R,E |
| 01161 | 04971 | R,E,L |
| 01161 | 05348 | R,E |
| 01161 | 09044 | R,E |
| 01161 | 09049 | R,E |
| 01161 | 09061 | R,E |
| 01162 | 00259 | E |
| 01162 | 00858 | R,E |
| 01162 | 00859 | R,E |
| 01162 | 00923 | R,E |
| 01162 | 00924 | R,E |
| 01162 | 01140 | R,E |
| 01162 | 01141 | R,E |
| 01162 | 01142 | R,E |
| 01162 | 01143 | R,E |
| 01162 | 01144 | R,E |
| 01162 | 01145 | R,E |
| 01162 | 01146 | R,E |
| 01162 | 01147 | R,E |
| 01162 | 01148 | R,E |
| 01162 | 01149 | R,E |
| 01162 | 01153 | R,E |
| 01162 | 01154 | R,E |
| 01162 | 01155 | R,E |
| 01162 | 01160 | R,E |
| 01162 | 04909 | R,E |
| 01162 | 04971 | R,E |
| 01162 | 05348 | R,E |
| 01162 | 09044 | R,E |
| 01162 | 09049 | R,E |
| 01162 | 09061 | R,E |
| 01163 | 00924 | R,E |
| 01163 | 01130 | R,E |
| 01163 | 01148 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01163 | 01164 | R,E |
| 01163 | 05354 | R,E |
| 01164 | 00819 | R,E |
| 01164 | 00858 | R,E |
| 01164 | 00859 | R,E |
| 01164 | 00923 | R,E |
| 01164 | 00924 | R,E |
| 01164 | 01140 | R,E |
| 01164 | 01148 | R,E |
| 01164 | 01163 | R,E |
| 01164 | 05348 | R,E |
| 01164 | 05354 | R,E |
| 01167 | 00915 | R,E |
| 01167 | 01025 | R,E |
| 01167 | 01251 | R,E |
| 01167 | 05347 | R,E |
| 01168 | 01123 | R,E |
| 01168 | 01124 | R,E |
| 01168 | 01251 | R,E |
| 01168 | 05347 | R,E |
| 01175 | 01155 | R |
| 01250 | 00037 | R,L |
| 01250 | 00259 | E |
| 01250 | 00273 | R,E,L |
| 01250 | 00367 | E |
| 01250 | 00500 | R,E,L |
| 01250 | 00819 | R |
| 01250 | 00850 | R |
| 01250 | 00852 | R |
| 01250 | 00855 | R |
| 01250 | 00870 | R,E,L |
| 01250 | 00912 | R |
| 01250 | 01252 | R |
| 01250 | 01282 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01250 | 05346 | E |
| 01251 | 00037 | R,L |
| 01251 | 00256 | R,E |
| 01251 | 00259 | E |
| 01251 | 00367 | E |
| 01251 | 00500 | R,E,L |
| 01251 | 00819 | R |
| 01251 | 00850 | R |
| 01251 | 00855 | R |
| 01251 | 00866 | R |
| 01251 | 00878 | R |
| 01251 | 00880 | R,E,L |
| 01251 | 00915 | R |
| 01251 | 01025 | R,E,L |
| 01251 | 01123 | R,E,L |
| 01251 | 01124 | R,E |
| 01251 | 01125 | R,E |
| 01251 | 01131 | R |
| 01251 | 01167 | R,E |
| 01251 | 01168 | R,E |
| 01251 | 01252 | R |
| 01251 | 01283 | R |
| 01251 | 05347 | E |
| 01252 | 00037 | R,E,L |
| 01252 | 00256 | R,E |
| 01252 | 00259 | E |
| 01252 | 00273 | R,E,L |
| 01252 | 00274 | R,E,L |
| 01252 | 00275 | R,E,L |
| 01252 | 00277 | R,E,L |
| 01252 | 00278 | R,E,L |
| 01252 | 00280 | R,E,L |
| 01252 | 00284 | R,E,L |
| 01252 | 00285 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 01252 | 00290 | E,L |
| 01252 | 00297 | R,E,L |
| 01252 | 00367 | E |
| 01252 | 00420 | R,L |
| 01252 | 00423 | R |
| 01252 | 00424 | R,L |
| 01252 | 00437 | R |
| 01252 | 00500 | R,E,L |
| 01252 | 00737 | R |
| 01252 | 00775 | R,E |
| 01252 | 00803 | R,E |
| 01252 | 00813 | R |
| 01252 | 00819 | R |
| 01252 | 00833 | E,L |
| 01252 | 00836 | E,L |
| 01252 | 00838 | E,L |
| 01252 | 00850 | R,E |
| 01252 | 00852 | R |
| 01252 | 00855 | R |
| 01252 | 00857 | R |
| 01252 | 00858 | R,E |
| 01252 | 00860 | R |
| 01252 | 00861 | R |
| 01252 | 00862 | R |
| 01252 | 00863 | R |
| 01252 | 00864 | R |
| 01252 | 00865 | R |
| 01252 | 00866 | R |
| 01252 | 00869 | R |
| 01252 | 00870 | R,L |
| 01252 | 00871 | R,E,L |
| 01252 | 00874 | E |
| 01252 | 00875 | R,L |
| 01252 | 00880 | R,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 01252 | 00897 | E |
| 01252 | 00903 | E |
| 01252 | 00905 | R |
| 01252 | 00912 | R |
| 01252 | 00914 | R |
| 01252 | 00915 | R |
| 01252 | 00916 | R |
| 01252 | 00920 | R |
| 01252 | 00921 | R |
| 01252 | 00922 | R |
| 01252 | 00923 | R,E |
| 01252 | 00924 | R,E,L |
| 01252 | 01025 | R,L |
| 01252 | 01026 | R,L |
| 01252 | 01027 | E,L |
| 01252 | 01041 | E |
| 01252 | 01047 | R,E,L |
| 01252 | 01051 | R |
| 01252 | 01097 | R |
| 01252 | 01098 | R |
| 01252 | 01112 | R,E,L |
| 01252 | 01122 | R,E,L |
| 01252 | 01123 | R,E,L |
| 01252 | 01126 | E |
| 01252 | 01130 | R,E |
| 01252 | 01132 | R,E |
| 01252 | 01137 | R,E |
| 01252 | 01140 | R,E,L |
| 01252 | 01141 | R,E,L |
| 01252 | 01142 | R,E,L |
| 01252 | 01143 | R,E,L |
| 01252 | 01144 | R,E,L |
| 01252 | 01145 | R,E,L |
| 01252 | 01146 | R,E,L |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01252 | 01147 | R,E,L |
| 01252 | 01148 | R,E,L |
| 01252 | 01149 | R,E,L |
| 01252 | 01250 | R |
| 01252 | 01251 | R |
| 01252 | 01254 | R |
| 01252 | 01255 | R |
| 01252 | 01257 | R |
| 01252 | 01275 | R |
| 01252 | 01280 | R |
| 01252 | 01281 | R |
| 01252 | 01283 | R |
| 01252 | 05348 | E |
| 01253 | 00037 | R,L |
| 01253 | 00259 | E |
| 01253 | 00367 | E |
| 01253 | 00423 | R,E |
| 01253 | 00500 | R,E,L |
| 01253 | 00737 | R,E |
| 01253 | 00813 | R |
| 01253 | 00819 | R |
| 01253 | 00850 | R |
| 01253 | 00869 | R |
| 01253 | 00875 | R,E,L |
| 01253 | 01280 | R |
| 01253 | 01287 | R,E |
| 01253 | 05349 | E |
| 01254 | 00037 | R,L |
| 01254 | 00259 | E |
| 01254 | 00367 | E |
| 01254 | 00500 | R,E,L |
| 01254 | 00819 | R |
| 01254 | 00850 | R |
| 01254 | 00857 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01254 | 00869 | R |
| 01254 | 00905 | R,E |
| 01254 | 00920 | R |
| 01254 | 01026 | R,E |
| 01254 | 01047 | R,E,L |
| 01254 | 01252 | R |
| 01254 | 01281 | R |
| 01254 | 01288 | R,E |
| 01254 | 05350 | E |
| 01255 | 00037 | R,L |
| 01255 | 00259 | E |
| 01255 | 00367 | E |
| 01255 | 00424 | R,E,L |
| 01255 | 00500 | R,E,L |
| 01255 | 00803 | R,E |
| 01255 | 00819 | R |
| 01255 | 00850 | R |
| 01255 | 00856 | R |
| 01255 | 00862 | R,E |
| 01255 | 00916 | R,E |
| 01255 | 01252 | R |
| 01255 | 01281 | R |
| 01255 | 05012 | R,E |
| 01255 | 05351 | E |
| 01256 | 00259 | E |
| 01256 | 00367 | E |
| 01256 | 00420 | C,L |
| 01256 | 00425 | E,L |
| 01256 | 00500 | R,E,L |
| 01256 | 00720 | C |
| 01256 | 00850 | R |
| 01256 | 00864 | E |
| 01256 | 01046 | E |
| 01256 | 01089 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01256 | 01127 | C |
| 01256 | 05352 | E |
| 01257 | 00037 | R |
| 01257 | 00259 | E |
| 01257 | 00367 | E |
| 01257 | 00437 | R |
| 01257 | 00500 | R |
| 01257 | 00775 | R |
| 01257 | 00819 | R |
| 01257 | 00850 | R |
| 01257 | 00914 | R |
| 01257 | 00921 | R,E |
| 01257 | 00922 | R,E |
| 01257 | 01112 | R,E |
| 01257 | 01122 | R,E |
| 01257 | 01252 | R |
| 01257 | 05353 | E |
| 01258 | 00037 | R,E |
| 01258 | 00259 | E |
| 01258 | 00500 | R,E |
| 01258 | 00819 | R |
| 01258 | 01129 | R,E |
| 01258 | 01130 | R,E |
| 01258 | 05354 | E |
| 01275 | 00037 | R |
| 01275 | 00256 | R |
| 01275 | 00273 | R |
| 01275 | 00277 | R |
| 01275 | 00278 | R |
| 01275 | 00280 | R |
| 01275 | 00284 | R |
| 01275 | 00285 | R |
| 01275 | 00297 | R |
| 01275 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01275 | 00437 | R |
| 01275 | 00500 | R |
| 01275 | 00819 | R |
| 01275 | 00850 | R |
| 01275 | 00858 | R,E |
| 01275 | 00863 | R |
| 01275 | 00871 | R |
| 01275 | 00923 | R,E |
| 01275 | 00924 | R,E |
| 01275 | 01051 | R |
| 01275 | 01140 | R,E |
| 01275 | 01141 | R,E |
| 01275 | 01142 | R,E |
| 01275 | 01143 | R,E |
| 01275 | 01144 | R,E |
| 01275 | 01145 | R,E |
| 01275 | 01146 | R,E |
| 01275 | 01147 | R,E |
| 01275 | 01148 | R,E |
| 01275 | 01149 | R,E |
| 01275 | 01252 | R |
| 01275 | 05348 | R,E |
| 01276 | 00367 | E |
| 01277 | 00367 | E |
| 01280 | 00037 | R |
| 01280 | 00367 | E |
| 01280 | 00423 | R |
| 01280 | 00437 | R |
| 01280 | 00500 | R |
| 01280 | 00737 | R,E |
| 01280 | 00813 | R |
| 01280 | 00819 | R |
| 01280 | 00850 | R |
| 01280 | 00869 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01280 | 00875 | R |
| 01280 | 01252 | R |
| 01280 | 01253 | R |
| 01280 | 01287 | R,E |
| 01280 | 04971 | R,E |
| 01280 | 05349 | R,E |
| 01281 | 00037 | R |
| 01281 | 00367 | E |
| 01281 | 00437 | R |
| 01281 | 00500 | R |
| 01281 | 00819 | R |
| 01281 | 00850 | R |
| 01281 | 00857 | R |
| 01281 | 00905 | R |
| 01281 | 00920 | R |
| 01281 | 01026 | R |
| 01281 | 01148 | R,E |
| 01281 | 01155 | R,E |
| 01281 | 01252 | R |
| 01281 | 01254 | R |
| 01281 | 01255 | R |
| 01281 | 05350 | R,E |
| 01282 | 00367 | E |
| 01282 | 00500 | R |
| 01282 | 00852 | R |
| 01282 | 00870 | R |
| 01282 | 00912 | R |
| 01282 | 01153 | R,E |
| 01282 | 01250 | R |
| 01282 | 05346 | R,E |
| 01283 | 00037 | R |
| 01283 | 00367 | E |
| 01283 | 00437 | R |
| 01283 | 00500 | R |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01283 | 00819 | R |
| 01283 | 00850 | R |
| 01283 | 00855 | R |
| 01283 | 00866 | R |
| 01283 | 00878 | R,E |
| 01283 | 00880 | R |
| 01283 | 00915 | R |
| 01283 | 01025 | R |
| 01283 | 01123 | R |
| 01283 | 01124 | R |
| 01283 | 01125 | R |
| 01283 | 01131 | R |
| 01283 | 01154 | R,E |
| 01283 | 01251 | R |
| 01283 | 01252 | R |
| 01283 | 05347 | R,E |
| 01284 | 05346 | R,E |
| 01285 | 05346 | R,E |
| 01287 | 00737 | R,E |
| 01287 | 00813 | R,E |
| 01287 | 00869 | R,E |
| 01287 | 00875 | R,E |
| 01287 | 01253 | R,E |
| 01287 | 01280 | R,E |
| 01287 | 05349 | R,E |
| 01288 | 00857 | R,E |
| 01288 | 00920 | R,E |
| 01288 | 01026 | R,E |
| 01288 | 01254 | R,E |
| 01351 | 00300 | E |
| 01351 | 00301 | E |
| 01351 | 00941 | E |
| 01362 | 00834 | E |
| 01362 | 00926 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 01362 | 00951 | E |
| 01362 | 00971 | E |
| 01362 | 04930 | E |
| 01374 | 61956 | R |
| 01380 | 00837 | E |
| 01380 | 00928 | E |
| 01380 | 01385 | E |
| 01380 | 04933 | E |
| 01380 | 13125 | E |
| 01382 | 00837 | E |
| 01385 | 00837 | E |
| 01385 | 00928 | E |
| 01385 | 01380 | E |
| 01385 | 04933 | E |
| 01385 | 13125 | E |
| 04133 | 00367 | E |
| 04369 | 00367 | E |
| 04371 | 00367 | E |
| 04373 | 00367 | E |
| 04374 | 00367 | E |
| 04376 | 00367 | E |
| 04378 | 00367 | E |
| 04380 | 00367 | E |
| 04381 | 00367 | E |
| 04386 | 00367 | E |
| 04516 | 00367 | E |
| 04519 | 00367 | E |
| 04520 | 00367 | E |
| 04533 | 00367 | E |
| 04596 | 00367 | E |
| 04899 | 00867 | R,E |
| 04899 | 01148 | R,E |
| 04899 | 05012 | R,E |
| 04899 | 05351 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 04899 | 09048 | R,E |
| 04899 | 12712 | R,E |
| 04909 | 00037 | R,E,L |
| 04909 | 00500 | R,E,L |
| 04909 | 00858 | R,E |
| 04909 | 00859 | R,E |
| 04909 | 00875 | R,E,L |
| 04909 | 00923 | R,E |
| 04909 | 00924 | R,E,L |
| 04909 | 01140 | R,E,L |
| 04909 | 01141 | R,E,L |
| 04909 | 01142 | R,E,L |
| 04909 | 01143 | R,E,L |
| 04909 | 01144 | R,E,L |
| 04909 | 01145 | R,E,L |
| 04909 | 01146 | R,E,L |
| 04909 | 01147 | R,E,L |
| 04909 | 01148 | R,E,L |
| 04909 | 01149 | R,E,L |
| 04909 | 01153 | R,E,L |
| 04909 | 01154 | R,E,L |
| 04909 | 01155 | R,E,L |
| 04909 | 01160 | R,E,L |
| 04909 | 01161 | R,E |
| 04909 | 01162 | R,E |
| 04909 | 04971 | R,E |
| 04909 | 05348 | R,E |
| 04909 | 05349 | R,E |
| 04909 | 09044 | R,E |
| 04909 | 09049 | R,E |
| 04909 | 09061 | R,E |
| 04929 | 00367 | E |
| 04930 | 00834 | E |
| 04930 | 00951 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 04930 | 01362 | E |
| 04932 | 00367 | E |
| 04933 | 00837 | E |
| 04933 | 01380 | E |
| 04933 | 01385 | E |
| 04934 | 00367 | E |
| 04946 | 00367 | E |
| 04946 | 00437 | E |
| 04947 | 00367 | E |
| 04949 | 00367 | E |
| 04953 | 00367 | E |
| 04953 | 00850 | E |
| 04964 | 00367 | E |
| 04965 | 00367 | E |
| 04966 | 00367 | E |
| 04967 | 00367 | E |
| 04970 | 00367 | E |
| 04970 | 00874 | E |
| 04971 | 00858 | R,E,M |
| 04971 | 00859 | R,E,M |
| 04971 | 00867 | R,E,M |
| 04971 | 00923 | R,E,L |
| 04971 | 00924 | R,E |
| 04971 | 01140 | R,E |
| 04971 | 01141 | R,E |
| 04971 | 01142 | R,E |
| 04971 | 01143 | R,E |
| 04971 | 01144 | R,E |
| 04971 | 01145 | R,E |
| 04971 | 01146 | R,E |
| 04971 | 01147 | R,E |
| 04971 | 01148 | R,E |
| 04971 | 01149 | R,E |
| 04971 | 01153 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 04971 | 01154 | R,E |
| 04971 | 01155 | R,E |
| 04971 | 01156 | R,E |
| 04971 | 01157 | R,E |
| 04971 | 01160 | R,E |
| 04971 | 01161 | R,E,L |
| 04971 | 01162 | R,E |
| 04971 | 01280 | R,E |
| 04971 | 04909 | R,E |
| 04971 | 05348 | R,E,L |
| 04971 | 05349 | R,E,L |
| 04971 | 09044 | R,E,L |
| 04971 | 09049 | R,E |
| 04971 | 09061 | R,E |
| 04971 | 17248 | R,E,L |
| 04976 | 00367 | E |
| 04992 | 00367 | E |
| 04993 | 00367 | E |
| 05012 | 00867 | R,E |
| 05012 | 01255 | R,E |
| 05012 | 04899 | R,E |
| 05012 | 12712 | R,E |
| 05014 | 00367 | E |
| 05100 | 00367 | E |
| 05104 | 16804 | R,E |
| 05104 | 17248 | R |
| 05123 | 00858 | E,L |
| 05123 | 01140 | E |
| 05123 | 01141 | E |
| 05123 | 01142 | E |
| 05123 | 01143 | E |
| 05123 | 01144 | E |
| 05123 | 01145 | E |
| 05123 | 01146 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 05123 | 01147 | E |
| 05123 | 01148 | E |
| 05123 | 01149 | E |
| 05123 | 01154 | E |
| 05123 | 01156 | E |
| 05123 | 01157 | E |
| 05123 | 01160 | E |
| 05123 | 05348 | E,L |
| 05123 | 08482 | R |
| 05137 | 00367 | E |
| 05143 | 00367 | E |
| 05210 | 00037 | E |
| 05210 | 00858 | R,E |
| 05210 | 00923 | R,E |
| 05210 | 01148 | R,E |
| 05210 | 01159 | E |
| 05210 | 05348 | E |
| 05211 | 00367 | E |
| 05346 | 00852 | R,E |
| 05346 | 00870 | R,E,L |
| 05346 | 00872 | R,E |
| 05346 | 00912 | R,E |
| 05346 | 01148 | R,E,L |
| 05346 | 01153 | R,E,L |
| 05346 | 01250 | E |
| 05346 | 01282 | R,E |
| 05346 | 01284 | R,E |
| 05346 | 01285 | R,E |
| 05346 | 09044 | R,E |
| 05347 | 00808 | R,E |
| 05347 | 00848 | R,E |
| 05347 | 00849 | R,E |
| 05347 | 00855 | R,E |
| 05347 | 00866 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 05347 | 00872 | R,E |
| 05347 | 00878 | R,E |
| 05347 | 00880 | R,E,L |
| 05347 | 00915 | R,E |
| 05347 | 01025 | R,E,L |
| 05347 | 01123 | R,E,L |
| 05347 | 01124 | R,E |
| 05347 | 01125 | R,E |
| 05347 | 01131 | R,E |
| 05347 | 01148 | R,E,L |
| 05347 | 01154 | R,E,L |
| 05347 | 01158 | R,E,L |
| 05347 | 01167 | R,E |
| 05347 | 01168 | R,E |
| 05347 | 01251 | E |
| 05347 | 01283 | R,E |
| 05348 | 00037 | R,E,L |
| 05348 | 00259 | E |
| 05348 | 00273 | R,E,L |
| 05348 | 00275 | R,E,L |
| 05348 | 00277 | R,E,L |
| 05348 | 00278 | R,E,L |
| 05348 | 00280 | R,E,L |
| 05348 | 00284 | R,E,L |
| 05348 | 00285 | R,E,L |
| 05348 | 00290 | E,L |
| 05348 | 00297 | R,E,L |
| 05348 | 00425 | R,E,L |
| 05348 | 00437 | R,E |
| 05348 | 00500 | R,E,L |
| 05348 | 00808 | R,E |
| 05348 | 00819 | R,E |
| 05348 | 00850 | R,E |
| 05348 | 00858 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 05348 | 00860 | R,E |
| 05348 | 00861 | R,E |
| 05348 | 00863 | R,E |
| 05348 | 00865 | R,E |
| 05348 | 00871 | R,E,L |
| 05348 | 00872 | R,E |
| 05348 | 00901 | R,E |
| 05348 | 00902 | R,E |
| 05348 | 00923 | R,E |
| 05348 | 00924 | R,E,L |
| 05348 | 01027 | E,L |
| 05348 | 01047 | R,L |
| 05348 | 01051 | R,E |
| 05348 | 01140 | R,E,L |
| 05348 | 01141 | R,E,L |
| 05348 | 01142 | R,E,L |
| 05348 | 01143 | R,E,L |
| 05348 | 01144 | R,E,L |
| 05348 | 01145 | R,E,L |
| 05348 | 01146 | R,E,L |
| 05348 | 01147 | R,E,L |
| 05348 | 01148 | R,E,L |
| 05348 | 01149 | R,E,L |
| 05348 | 01153 | R,E,L |
| 05348 | 01154 | R,E,L |
| 05348 | 01155 | R,E,L |
| 05348 | 01156 | R,E,L |
| 05348 | 01157 | R,E,L |
| 05348 | 01158 | R,E,L |
| 05348 | 01160 | R,E,L |
| 05348 | 01161 | R,E |
| 05348 | 01162 | R,E |
| 05348 | 01164 | R,E |
| 05348 | 01252 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 05348 | 01275 | R,E |
| 05348 | 04909 | R,E |
| 05348 | 04971 | R,E,L |
| 05348 | 05123 | E,L |
| 05348 | 05210 | E |
| 05348 | 08482 | E,L |
| 05348 | 09044 | R,E |
| 05348 | 09049 | R,E |
| 05348 | 09061 | R,E |
| 05348 | 12712 | R,E,L |
| 05348 | 16804 | R,E,L |
| 05348 | 17248 | R,E |
| 05349 | 00813 | R,E |
| 05349 | 00869 | R,E |
| 05349 | 00875 | R,E,L |
| 05349 | 01148 | R,E,L |
| 05349 | 01253 | E |
| 05349 | 01280 | R,E |
| 05349 | 01287 | R,E |
| 05349 | 04909 | R,E |
| 05349 | 04971 | R,E,L |
| 05349 | 09061 | R,E |
| 05350 | 00500 | R,L |
| 05350 | 00857 | R,E |
| 05350 | 00920 | R,E |
| 05350 | 01026 | R,E,L |
| 05350 | 01148 | R,E,L |
| 05350 | 01155 | R,E,L |
| 05350 | 01254 | E |
| 05350 | 01281 | R,E |
| 05350 | 09049 | R,E |
| 05350 | 09061 | R,E |
| 05351 | 00424 | R,E,L |
| 05351 | 00856 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 05351 | 00862 | R,E |
| 05351 | 00867 | R,E |
| 05351 | 00916 | R,E |
| 05351 | 01148 | R,E,L |
| 05351 | 01255 | E |
| 05351 | 04899 | R,E |
| 05351 | 09048 | R,E |
| 05351 | 12712 | R,E,L |
| 05352 | 00420 | C,L |
| 05352 | 00425 | R,E,L |
| 05352 | 00864 | E |
| 05352 | 01046 | E |
| 05352 | 01089 | E |
| 05352 | 01148 | R,E,L |
| 05352 | 01256 | E |
| 05352 | 09238 | E |
| 05352 | 16804 | C,L |
| 05352 | 17248 | E |
| 05353 | 00901 | R,E |
| 05353 | 00902 | R,E |
| 05353 | 00921 | R,E |
| 05353 | 00922 | R,E |
| 05353 | 01112 | R,E |
| 05353 | 01122 | R,E |
| 05353 | 01148 | R,E |
| 05353 | 01156 | R,E |
| 05353 | 01157 | R,E |
| 05353 | 01257 | E |
| 05354 | 01129 | R,E |
| 05354 | 01130 | R,E |
| 05354 | 01148 | R,E |
| 05354 | 01163 | R,E |
| 05354 | 01164 | R,E |
| 05354 | 01258 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|-------------------|-----------------|----------------------------|
| 05470 | 61956 | R |
| 08229 | 00367 | E |
| 08448 | 00367 | E |
| 08482 | 00858 | R,E,M |
| 08482 | 01148 | R,E |
| 08482 | 05123 | R |
| 08482 | 05348 | E,L |
| 08629 | 00367 | E |
| 08692 | 00367 | E |
| 09025 | 00367 | E |
| 09027 | 21427 | E |
| 09028 | 00367 | E |
| 09044 | 00858 | R,E |
| 09044 | 00859 | R,E |
| 09044 | 00870 | R,E,L |
| 09044 | 00872 | R,E |
| 09044 | 00923 | R,E |
| 09044 | 00924 | R,E,L |
| 09044 | 01140 | R,E,L |
| 09044 | 01141 | R,E,L |
| 09044 | 01142 | R,E,L |
| 09044 | 01143 | R,E,L |
| 09044 | 01144 | R,E,L |
| 09044 | 01145 | R,E,L |
| 09044 | 01146 | R,E,L |
| 09044 | 01147 | R,E,L |
| 09044 | 01148 | R,E,L |
| 09044 | 01149 | R,E,L |
| 09044 | 01153 | R,E,L |
| 09044 | 01154 | R,E,L |
| 09044 | 01155 | R,E,L |
| 09044 | 01160 | R,E,L |
| 09044 | 01161 | R,E |
| 09044 | 01162 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 09044 | 04909 | R,E |
| 09044 | 04971 | R,E,L |
| 09044 | 05346 | R,E |
| 09044 | 05348 | R,E |
| 09044 | 09049 | R,E |
| 09044 | 09061 | R,E |
| 09044 | 12712 | R,E,L |
| 09044 | 16804 | R,E,L |
| 09047 | 00367 | E |
| 09048 | 00424 | R |
| 09048 | 00867 | R,E |
| 09048 | 00916 | R,E |
| 09048 | 01148 | R,E |
| 09048 | 04899 | R,E |
| 09048 | 05351 | R,E |
| 09048 | 12712 | R,E |
| 09049 | 00500 | E |
| 09049 | 00858 | R,E |
| 09049 | 00859 | R,E |
| 09049 | 00872 | R,E |
| 09049 | 00923 | R,E |
| 09049 | 00924 | R,E |
| 09049 | 01026 | R |
| 09049 | 01140 | R,E |
| 09049 | 01141 | R,E |
| 09049 | 01142 | R,E |
| 09049 | 01143 | R,E |
| 09049 | 01144 | R,E |
| 09049 | 01145 | R,E |
| 09049 | 01146 | R,E |
| 09049 | 01147 | R,E |
| 09049 | 01148 | R,E |
| 09049 | 01149 | R,E |
| 09049 | 01153 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 09049 | 01154 | R,E |
| 09049 | 01155 | R,E |
| 09049 | 01160 | R,E |
| 09049 | 01161 | R,E |
| 09049 | 01162 | R,E |
| 09049 | 04909 | R,E |
| 09049 | 04971 | R,E |
| 09049 | 05348 | R,E |
| 09049 | 05350 | R,E |
| 09049 | 09044 | R,E |
| 09049 | 09061 | R,E |
| 09049 | 12712 | R,E |
| 09049 | 16804 | R,E |
| 09060 | 00367 | E |
| 09061 | 00423 | R,E |
| 09061 | 00858 | R,E |
| 09061 | 00859 | R,E |
| 09061 | 00875 | R,E |
| 09061 | 00923 | R,E |
| 09061 | 00924 | R,E |
| 09061 | 01140 | R,E |
| 09061 | 01141 | R,E |
| 09061 | 01142 | R,E |
| 09061 | 01143 | R,E |
| 09061 | 01144 | R,E |
| 09061 | 01145 | R,E |
| 09061 | 01146 | R,E |
| 09061 | 01147 | R,E |
| 09061 | 01148 | R,E |
| 09061 | 01149 | R,E |
| 09061 | 01153 | R,E |
| 09061 | 01154 | R,E |
| 09061 | 01155 | R,E |
| 09061 | 01160 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 09061 | 01161 | R,E |
| 09061 | 01162 | R,E |
| 09061 | 04909 | R,E |
| 09061 | 04971 | R,E |
| 09061 | 05348 | R,E |
| 09061 | 05349 | R,E |
| 09061 | 05350 | R,E |
| 09061 | 09044 | R,E |
| 09061 | 09049 | R,E |
| 09089 | 00367 | E |
| 09238 | 00420 | E,L |
| 09238 | 00924 | R,E,L |
| 09238 | 01089 | C |
| 09238 | 01148 | R,E,L |
| 09238 | 05352 | E |
| 09238 | 16804 | C,L |
| 09238 | 17248 | C |
| 09444 | 61956 | R |
| 09447 | 12712 | R,E |
| 12544 | 00367 | E |
| 12712 | 00862 | R,E,M |
| 12712 | 00867 | R,E,M |
| 12712 | 00916 | R,E,L |
| 12712 | 01148 | R,E |
| 12712 | 01156 | R,E |
| 12712 | 01157 | R,E |
| 12712 | 04899 | R,E |
| 12712 | 05012 | R,E |
| 12712 | 05348 | R,E,L |
| 12712 | 05351 | R,E,L |
| 12712 | 09044 | R,E,L |
| 12712 | 09048 | R,E |
| 12712 | 09049 | R,E |
| 12712 | 09447 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 12712 | 16804 | R,E |
| 12712 | 17248 | R,E,L |
| 12725 | 00367 | E |
| 12788 | 00367 | E |
| 13121 | 01126 | E,L |
| 13125 | 00837 | E |
| 13125 | 01380 | E |
| 13125 | 01385 | E |
| 13152 | 00367 | E |
| 16421 | 00367 | E |
| 16804 | 00425 | C |
| 16804 | 00858 | R,E,M |
| 16804 | 00859 | R,E,M |
| 16804 | 00867 | R,E,M |
| 16804 | 00923 | R,E,L |
| 16804 | 00924 | R,E |
| 16804 | 01140 | R,E |
| 16804 | 01148 | R,E |
| 16804 | 01154 | R,E |
| 16804 | 01155 | R,E |
| 16804 | 01156 | R,E |
| 16804 | 01157 | R,E |
| 16804 | 05104 | R,E |
| 16804 | 05348 | R,E,L |
| 16804 | 05352 | C,L |
| 16804 | 09044 | R,E,L |
| 16804 | 09049 | R,E |
| 16804 | 09238 | C,L |
| 16804 | 12712 | R,E |
| 16804 | 17248 | R,E,L |
| 16821 | 00367 | E |
| 16884 | 00367 | E |
| 17248 | 00420 | R,E,L |
| 17248 | 00858 | R,E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 17248 | 00859 | R,E |
| 17248 | 00923 | R,E |
| 17248 | 00924 | R,E,L |
| 17248 | 01140 | R,E,L |
| 17248 | 01141 | R,E,L |
| 17248 | 01142 | R,E,L |
| 17248 | 01143 | R,E,L |
| 17248 | 01144 | R,E,L |
| 17248 | 01145 | R,E,L |
| 17248 | 01146 | R,E,L |
| 17248 | 01147 | R,E,L |
| 17248 | 01148 | R,E,L |
| 17248 | 01149 | R,E,L |
| 17248 | 01153 | R,E,L |
| 17248 | 01154 | R,E,L |
| 17248 | 01155 | R,E,L |
| 17248 | 01160 | R,E,L |
| 17248 | 04971 | R,E,L |
| 17248 | 05104 | R |
| 17248 | 05348 | R,E |
| 17248 | 05352 | E |
| 17248 | 09238 | C |
| 17248 | 12712 | R,E,L |
| 17248 | 16804 | R,E,L |
| 20517 | 00367 | E |
| 20917 | 00367 | E |
| 20980 | 00367 | E |
| 21427 | 09027 | E |
| 24613 | 00367 | E |
| 25013 | 00367 | E |
| 25076 | 00367 | E |
| 25426 | 00367 | E |
| 25427 | 00367 | E |
| 25428 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 25429 | 00367 | E |
| 25431 | 00367 | E |
| 25432 | 00367 | E |
| 25433 | 00367 | E |
| 25436 | 00367 | E |
| 25437 | 00367 | E |
| 25438 | 00367 | E |
| 25439 | 00367 | E |
| 25440 | 00367 | E |
| 25441 | 00367 | E |
| 25442 | 00367 | E |
| 25444 | 00367 | E |
| 25445 | 00367 | E |
| 25450 | 00367 | E |
| 25467 | 00367 | E |
| 25473 | 00367 | E |
| 25479 | 00367 | E |
| 25480 | 00367 | E |
| 25580 | 00367 | E |
| 25616 | 00367 | E |
| 25617 | 00367 | E |
| 25618 | 00367 | E |
| 25619 | 00367 | E |
| 25664 | 00367 | E |
| 25690 | 00367 | E |
| 25691 | 00367 | E |
| 28709 | 00437 | R,E,M |
| 28709 | 00737 | E |
| 28709 | 00775 | E |
| 28709 | 00852 | R,E,M |
| 28709 | 00857 | R,E |
| 28709 | 00860 | R,E |
| 28709 | 00861 | R,E,M |
| 28709 | 00862 | R,E,M |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 28709 | 00863 | R,E |
| 28709 | 00864 | R,E,M |
| 28709 | 00865 | R,E |
| 28709 | 01114 | E |
| 29109 | 00367 | E |
| 29172 | 00367 | E |
| 29522 | 00367 | E |
| 29523 | 00367 | E |
| 29524 | 00367 | E |
| 29525 | 00367 | E |
| 29527 | 00367 | E |
| 29528 | 00367 | E |
| 29529 | 00367 | E |
| 29532 | 00367 | E |
| 29533 | 00367 | E |
| 29534 | 00367 | E |
| 29535 | 00367 | E |
| 29536 | 00367 | E |
| 29537 | 00367 | E |
| 29540 | 00367 | E |
| 29541 | 00367 | E |
| 29546 | 00367 | E |
| 29712 | 00367 | E |
| 29713 | 00367 | E |
| 29714 | 00367 | E |
| 29715 | 00367 | E |
| 29760 | 00367 | E |
| 32805 | 00367 | E |
| 33058 | 00367 | E |
| 33205 | 00367 | E |
| 33268 | 00367 | E |
| 33618 | 00367 | E |
| 33619 | 00367 | E |
| 33620 | 00367 | E |

Table 61. Non-Unicode Conversions Available (continued)

| FROM-CCSID | TO-CCSID | Technique Supported |
|------------|----------|---------------------|
| 33621 | 00367 | E |
| 33623 | 00367 | E |
| 33624 | 00367 | E |
| 33632 | 00367 | E |
| 33636 | 00367 | E |
| 33637 | 00367 | E |
| 33665 | 00367 | E |
| 37301 | 00367 | E |
| 37719 | 00367 | E |
| 37728 | 00367 | E |
| 37732 | 00367 | E |
| 37761 | 00367 | E |
| 41397 | 00367 | E |
| 41460 | 00367 | E |
| 41824 | 00367 | E |
| 41828 | 00367 | E |
| 45493 | 00367 | E |
| 45556 | 00367 | E |
| 45920 | 00367 | E |
| 49589 | 00367 | E |
| 49652 | 00367 | E |
| 53748 | 00367 | E |
| 61696 | 00367 | E |
| 61697 | 00367 | E |
| 61698 | 00367 | E |
| 61699 | 00367 | E |
| 61710 | 00367 | E |
| 61711 | 00367 | E |
| 61712 | 00367 | E |
| 61956 | 01374 | E |
| 61956 | 05470 | E |
| 61956 | 09444 | E |

Direct conversions supported to and from Unicode

The following table lists direct conversions supported between non-Unicode CCSIDs and Unicode CCSID 01200. (CCSID 01200 is the "virtual" CCSID for UTF-16. A specific UTF-16 CCSID is substituted for 01200, such as 13488, 17584, 21680, 42160, or 54448.) The specific Unicode CCSID supported is shown for each conversion.

Note: Each CCSID may be supported by more than one level of Unicode. Also, conversions between Unicode CCSIDs 01200, 01208 and 01232 are supported by algorithmic conversions.

| <i>Table 62. Direct Conversions Supported to and from Unicode CCSID 01200</i> | | | |
|---|---|--|----------------------|
| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
| 00037 | R,L | E,L | 13488 |
| 00256 | R | E | 13488 |
| 00259 | R,E | E | 13488 |
| 00273 | R,L | E,L | 13488 |
| 00274 | R,L | E,L | 17584 |
| 00275 | R,L | E,L | 13488 |
| 00277 | R,L | E,L | 13488 |
| 00278 | R,L | E,L | 13488 |
| 00280 | R,L | E,L | 13488 |
| 00282 | R,L | E,L | 13488 |
| 00284 | R,L | E,L | 13488 |
| 00285 | R,L | E,L | 13488 |
| 00286 | R | E | 17584 |
| 00290 | R,L | E,C,L | 13488 |
| 00293 | R,E | E | 13488 |
| 00297 | R,L | E,L | 13488 |
| 00300 | R | E | 13488 |
| 00301 | R | E | 13488 |
| 00367 | R | E,C | 13488 |
| 00420 | R,C,L | E,C,L | 13488 |
| 00423 | R | E | 13488 |
| 00424 | R,L | E,L | 13488 |
| 00425 | R,L | E,L | 17584 |
| 00437 | R | E | 13488 |
| 00500 | R,L | E,L | 13488 |
| 00720 | R | E | 13488 |
| 00737 | R | E | 13488 |
| 00775 | R | E | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 00803 | R,L | R | 13488 |
| 00806 | R | E | 13488 |
| 00808 | R | E | 17584 |
| 00813 | R | E | 13488 |
| 00819 | R | E | 13488 |
| 00833 | R,L | E,C,L | 13488 |
| 00834 | R,E | E | 13488 |
| 00835 | E | E | 13488 |
| 00836 | R,L | E,C,L | 13488 |
| 00837 | R,E | E | 13488 |
| 00838 | E,L | E,L | 13488 |
| 00848 | R | E | 17584 |
| 00849 | R | E | 17584 |
| 00850 | R | E,M | 13488 |
| 00851 | R | E | 13488 |
| 00852 | R | E | 13488 |
| 00853 | R | E | 13488 |
| 00855 | R | E | 13488 |
| 00856 | R | E | 13488 |
| 00857 | R | E | 13488 |
| 00858 | R | E | 17584 |
| 00859 | R | E | 17584 |
| 00860 | R | E | 13488 |
| 00861 | R | E | 13488 |
| 00862 | R | E | 13488 |
| 00863 | R | E | 13488 |
| 00864 | R,C | E,C,M | 13488 |
| 00865 | R | E | 13488 |
| 00866 | R | E | 13488 |
| 00867 | R | E,M | 17584 |
| 00868 | R,E | E | 13488 |
| 00869 | R | E | 13488 |
| 00870 | R,L | E,L | 13488 |
| 00871 | R,L | E,L | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 00872 | R | E | 17584 |
| 00874 | E | E,M | 13488 |
| 00875 | R,L | E,L | 13488 |
| 00876 | R | E | 17584 |
| 00878 | R | E | 13488 |
| 00880 | R,L | E,L | 13488 |
| 00891 | R | E,C | 13488 |
| 00892 | R,L | E,L | 17584 |
| 00895 | R,M | E,C,M | 13488 |
| 00896 | R,M | E,M | 13488 |
| 00897 | R,M | E,C,M | 13488 |
| 00901 | R | E | 17584 |
| 00902 | R | E | 17584 |
| 00903 | R | E,C | 13488 |
| 00904 | R | E,C,M | 13488 |
| 00905 | R | E | 13488 |
| 00912 | R | E | 13488 |
| 00913 | R | E | 17584 |
| 00914 | R | E | 13488 |
| 00915 | R | E | 13488 |
| 00916 | R | E | 13488 |
| 00918 | R,E | E | 13488 |
| 00920 | R | E | 13488 |
| 00921 | R | E | 13488 |
| 00922 | R | E | 13488 |
| 00923 | R | E | 17584 |
| 00924 | R,L | E,L | 17584 |
| 00926 | R | E | 17584 |
| 00927 | E | E | 13488 |
| 00928 | R | E | 13488 |
| 00941 | E | E | 13488 |
| 00947 | E | E | 13488 |
| 00951 | R,E | E | 13488 |
| 00952 | E | E | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 00953 | E | E | 17584 |
| 00955 | E | E | 13488 |
| 00960 | E | E | 17584 |
| 00961 | R | E | 13488 |
| 00963 | E | E | 13488 |
| 00971 | R | E | 13488 |
| 01004 | R | E | 13488 |
| 01006 | R,E | E | 13488 |
| 01008 | R | E | 13488 |
| 01009 | R | E | 13488 |
| 01010 | R | E | 13488 |
| 01011 | R | E | 13488 |
| 01012 | R | E | 13488 |
| 01013 | R | E | 13488 |
| 01014 | R | E | 13488 |
| 01015 | R | E | 13488 |
| 01016 | R | E | 13488 |
| 01017 | R | E | 13488 |
| 01018 | R | E | 13488 |
| 01019 | R | E | 13488 |
| 01020 | R | E | 17584 |
| 01021 | R | E | 17584 |
| 01023 | R | E | 17584 |
| 01025 | R,L | E,L | 13488 |
| 01026 | R,L | E,L | 13488 |
| 01027 | R,L | E,C,L | 13488 |
| 01040 | R | E,C | 13488 |
| 01041 | R,M | E,C,M | 13488 |
| 01042 | R | E,C | 13488 |
| 01043 | R | E,C | 13488 |
| 01046 | R | E | 13488 |
| 01047 | R,L | E,L | 13488 |
| 01051 | E | E | 13488 |
| 01088 | R,C,L,M | E,C,M | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 01089 | R | E | 13488 |
| 01097 | R,E | E | 13488 |
| 01098 | R,E | E | 13488 |
| 01100 | R | E | 17584 |
| 01101 | R | E | 17584 |
| 01102 | R | E | 17584 |
| 01103 | R | E | 17584 |
| 01104 | R | E | 17584 |
| 01105 | R | E | 17584 |
| 01106 | R | E | 17584 |
| 01107 | R | E | 17584 |
| 01112 | R,L | E,L | 13488 |
| 01114 | R | E,C | 13488 |
| 01115 | R | E,C,M | 13488 |
| 01122 | R,L | E,L | 13488 |
| 01123 | R,L | E,L | 13488 |
| 01124 | R | E | 13488 |
| 01125 | R | E | 13488 |
| 01126 | R,M | E,M | 13488 |
| 01126 | R | E,M | 17584 |
| 01129 | R | E | 13488 |
| 01130 | R | E | 13488 |
| 01131 | R | E | 13488 |
| 01132 | R | E | 13488 |
| 01132 | R | E | 17584 |
| 01133 | E | E | 13488 |
| 01137 | R | E | 13488 |
| 01140 | R,L | E,L | 17584 |
| 01141 | R,L | E,L | 17584 |
| 01142 | R,L | E,L | 17584 |
| 01143 | R,L | E,L | 17584 |
| 01144 | R,L | E,L | 17584 |
| 01145 | R,L | E,L | 17584 |
| 01146 | R,L | E,L | 17584 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 01147 | R,L | E,L | 17584 |
| 01148 | R,L | E,L | 17584 |
| 01149 | R,L | E,L | 17584 |
| 01153 | R,L | E,L | 17584 |
| 01154 | R,L | E,L | 17584 |
| 01155 | R,L | E,L | 17584 |
| 01156 | R,L | E,L | 17584 |
| 01157 | R,L | E,L | 17584 |
| 01158 | R,L | E,L | 17584 |
| 01159 | R,L | E,C,L | 17584 |
| 01159 | R,L | E,C,L | 42160 |
| 01160 | R,L | E,L | 17584 |
| 01161 | R | E | 17584 |
| 01163 | R | E | 17584 |
| 01164 | R | E | 17584 |
| 01165 | R,L | E,L | 17584 |
| 01166 | R | E | 17584 |
| 01167 | R | E | 17584 |
| 01168 | R | E | 17584 |
| 01175 | R | E | 21680 |
| 01175 | R | E | 42160 |
| 01250 | R | E | 13488 |
| 01251 | R | E | 13488 |
| 01252 | R | E | 13488 |
| 01253 | R | E | 13488 |
| 01254 | R | E | 13488 |
| 01255 | R | E | 13488 |
| 01256 | R | E | 13488 |
| 01257 | R | E | 13488 |
| 01258 | R | E | 13488 |
| 01275 | R | E | 13488 |
| 01276 | R | E | 13488 |
| 01277 | E | E | 13488 |
| 01280 | R | E | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 01281 | R | E | 13488 |
| 01282 | R | E | 13488 |
| 01283 | R | E | 13488 |
| 01284 | R | E | 13488 |
| 01285 | R | E | 13488 |
| 01351 | R | E | 13488 |
| 01362 | R | E | 13488 |
| 01362 | R | E | 17584 |
| 01374 | R | E | 17584 |
| 01376 | R | E | 17584 |
| 01376 | R,C | E,C | 21680 |
| 01378 | R | E | 42160 |
| 01380 | R,E | E | 13488 |
| 01382 | R,E | E | 13488 |
| 01385 | R | E | 13488 |
| 01385 | R | E | 17584 |
| 01391 | C | C | 21680 |
| 04133 | R | E | 13488 |
| 04369 | R | E | 13488 |
| 04370 | R | E | 17584 |
| 04371 | R | E | 13488 |
| 04373 | R | E | 13488 |
| 04374 | R | E | 13488 |
| 04376 | R | E | 13488 |
| 04378 | R | E | 13488 |
| 04380 | R | E | 13488 |
| 04381 | R | E | 13488 |
| 04386 | R,C | E,C | 13488 |
| 04393 | R | E | 13488 |
| 04396 | R | E | 13488 |
| 04396 | R | E | 17584 |
| 04397 | R | E | 13488 |
| 04516 | R,C | E,C | 13488 |
| 04517 | C | C | 21680 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 04519 | R | E | 13488 |
| 04520 | R | E | 13488 |
| 04533 | R | E | 13488 |
| 04596 | R | E | 13488 |
| 04899 | R | E | 17584 |
| 04904 | R | E | 17584 |
| 04909 | R | E | 17584 |
| 04929 | R | E,C | 13488 |
| 04930 | R | E | 13488 |
| 04930 | R | E | 17584 |
| 04931 | E | E | 13488 |
| 04932 | R | E,C | 13488 |
| 04933 | R | E | 13488 |
| 04933 | R | E | 17584 |
| 04933 | R | E | 21680 |
| 04934 | E | E | 13488 |
| 04944 | R | E | 17584 |
| 04945 | R | E | 17584 |
| 04946 | R | E | 13488 |
| 04947 | R | E | 13488 |
| 04948 | R | E | 13488 |
| 04949 | R | E | 13488 |
| 04951 | R | E | 13488 |
| 04952 | R | E | 13488 |
| 04953 | R | E | 13488 |
| 04954 | R | E | 17584 |
| 04955 | R | E | 17584 |
| 04956 | R | E | 17584 |
| 04957 | R | E | 17584 |
| 04958 | R | E | 17584 |
| 04959 | R | E | 17584 |
| 04960 | R | E | 13488 |
| 04961 | R | E | 17584 |
| 04962 | R | E | 17584 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 04963 | R | E | 17584 |
| 04964 | R,E | E | 13488 |
| 04965 | R | E | 13488 |
| 04966 | R | E | 13488 |
| 04967 | R | E | 13488 |
| 04970 | E | E | 13488 |
| 04971 | R,L | E,L | 17584 |
| 04976 | R | E | 13488 |
| 04992 | R,M | E,M | 13488 |
| 04993 | R,M | E,C,M | 13488 |
| 05012 | R | E | 13488 |
| 05014 | R,E | E | 13488 |
| 05023 | E | E | 13488 |
| 05043 | E | E | 13488 |
| 05047 | R | E | 13488 |
| 05048 | E | E | 13488 |
| 05049 | E | E | 13488 |
| 05056 | R,E | E | 17584 |
| 05067 | E | E | 13488 |
| 05100 | R | E | 13488 |
| 05104 | R | E | 17584 |
| 05123 | R,L | E,C,L | 17584 |
| 05137 | R,M | E,C,M | 13488 |
| 05142 | R | E | 13488 |
| 05143 | R | E | 13488 |
| 05210 | R | E,C | 17584 |
| 05211 | R | E,C | 13488 |
| 05233 | R,L | E,L | 21680 |
| 05233 | R,L | E,L | 42160 |
| 05255 | R,L | E,L | 42160 |
| 05346 | R | E | 17584 |
| 05347 | R | E | 17584 |
| 05348 | R | E | 17584 |
| 05349 | R | E | 17584 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 05350 | R | E | 17584 |
| 05351 | R | E | 17584 |
| 05352 | R | E | 17584 |
| 05353 | R | E | 17584 |
| 05354 | R | E | 17584 |
| 05470 | R | E | 17584 |
| 05472 | R | E | 17584 |
| 05474 | R | E | 42160 |
| 05476 | R,E | E | 13488 |
| 05478 | R | E | 13488 |
| 05487 | C | C | 17584 |
| 08229 | R,C | E,C | 13488 |
| 08448 | R | E | 13488 |
| 08482 | R,L | E,C,L | 17584 |
| 08492 | R | E | 13488 |
| 08493 | R | E | 13488 |
| 08612 | R | E | 13488 |
| 08629 | R | E | 13488 |
| 08692 | R | E | 13488 |
| 09025 | R | E,C | 13488 |
| 09026 | R,E | E | 13488 |
| 09027 | E | E | 17584 |
| 09027 | R | E | 42160 |
| 09028 | R | E,C | 13488 |
| 09030 | E | E | 13488 |
| 09042 | R | E | 17584 |
| 09044 | R | E | 17584 |
| 09047 | R | E | 13488 |
| 09048 | R | E | 17584 |
| 09049 | R | E | 17584 |
| 09056 | R | E | 13488 |
| 09060 | R,E | E | 13488 |
| 09061 | R | E | 17584 |
| 09064 | R | E | 17584 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 09066 | E | E | 13488 |
| 09088 | R,E,M | R,E,C,M | 13488 |
| 09089 | R,M | E,C,M | 13488 |
| 09139 | E | E | 13488 |
| 09144 | R | E | 13488 |
| 09145 | R,E | E | 13488 |
| 09163 | R | E | 13488 |
| 09219 | L | C,L | 17584 |
| 09238 | R | E | 17584 |
| 09306 | R | E | 17584 |
| 09444 | R,C | E,C | 17584 |
| 09444 | C | E,C | 21680 |
| 09447 | R | E | 17584 |
| 09448 | C | C | 21680 |
| 09449 | R | E | 17584 |
| 09566 | R | E | 17584 |
| 09568 | R | E | 17584 |
| 09572 | R,E | E | 13488 |
| 09574 | R | E | 13488 |
| 09577 | C | C | 17584 |
| 09577 | C | C | 21680 |
| 12544 | R | E | 13488 |
| 12578 | L | C,L | 17584 |
| 12588 | R | E | 13488 |
| 12712 | R,L | E,L | 17584 |
| 12725 | R | E | 13488 |
| 12788 | R | E | 13488 |
| 13121 | R,L | E,C,L | 17584 |
| 13124 | R,L | E,C,L | 13488 |
| 13125 | R | E | 13488 |
| 13140 | R | E | 17584 |
| 13143 | R | E | 17584 |
| 13145 | R | E | 17584 |
| 13152 | R | E | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 13156 | R | E | 17584 |
| 13157 | R | E | 17584 |
| 13162 | R | E | 17584 |
| 13184 | R,M | C,M | 13488 |
| 13185 | R,C,M | M | 13488 |
| 13235 | E | E | 13488 |
| 13240 | R | E | 13488 |
| 13241 | R | E | 13488 |
| 13241 | R | E | 17584 |
| 13662 | R | E | 17584 |
| 13662 | R,C | E,C | 21680 |
| 13664 | R | E | 17584 |
| 13664 | R,C | E,C | 21680 |
| 16421 | R | E | 13488 |
| 16684 | R | E | 13488 |
| 16684 | R | E | 17584 |
| 16684 | R | E | 21680 |
| 16804 | R,L | E,L | 17584 |
| 16821 | R | E | 13488 |
| 16884 | R | E | 13488 |
| 17219 | R | E | 42160 |
| 17221 | R | E | 17584 |
| 17240 | R | E | 17584 |
| 17248 | R | E | 17584 |
| 17331 | E | E | 13488 |
| 17337 | R | E | 17584 |
| 20517 | R | E | 13488 |
| 20780 | R | E | 17584 |
| 20917 | R | E | 13488 |
| 20980 | R | E | 13488 |
| 21314 | R,E | E | 13488 |
| 21317 | R,E | E | 13488 |
| 21344 | R | E | 17584 |
| 21427 | E | E | 17584 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 21433 | R | E | 13488 |
| 24613 | R | E | 13488 |
| 24876 | R | E | 21680 |
| 24877 | R | E | 13488 |
| 25013 | R | E | 13488 |
| 25076 | R | E | 13488 |
| 25426 | R | E | 13488 |
| 25427 | R | E | 13488 |
| 25428 | R | E | 13488 |
| 25429 | R | E | 13488 |
| 25431 | R | E | 13488 |
| 25432 | R | E | 13488 |
| 25433 | R | E | 13488 |
| 25436 | R | E | 13488 |
| 25437 | R | E | 13488 |
| 25438 | R | E | 13488 |
| 25439 | R | E | 13488 |
| 25440 | R | E | 13488 |
| 25441 | R | E | 13488 |
| 25442 | R | E | 13488 |
| 25444 | R,E | E | 13488 |
| 25445 | R,E | E | 13488 |
| 25450 | E | E | 13488 |
| 25467 | R | E,C | 13488 |
| 25473 | R,M | E,C,M | 13488 |
| 25479 | R | E,C | 13488 |
| 25480 | R | E,C | 13488 |
| 25502 | R | E | 17584 |
| 25503 | E | E | 13488 |
| 25504 | R | E | 13488 |
| 25527 | R,E | E | 13488 |
| 25528 | R | E | 13488 |
| 25580 | R | E | 13488 |
| 25616 | R | E,C | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 25617 | R,M | E,C,M | 13488 |
| 25618 | R | E,C | 13488 |
| 25619 | R | E,C | 13488 |
| 25664 | R,M | E,C,M | 13488 |
| 25690 | R | E,C | 13488 |
| 25691 | R | E,C | 13488 |
| 28709 | R,L | E,C,L | 13488 |
| 28709 | R | E,C,L | 17584 |
| 28709 | R | E,C,L | 21680 |
| 28972 | R,C | E,C | 21680 |
| 29109 | R | E | 13488 |
| 29172 | R | E | 13488 |
| 29509 | R | E | 21680 |
| 29522 | R | E | 13488 |
| 29523 | R | E | 13488 |
| 29524 | R | E | 13488 |
| 29525 | R | E | 13488 |
| 29527 | R | E | 13488 |
| 29528 | R | E | 13488 |
| 29529 | R | E | 13488 |
| 29532 | R | E | 13488 |
| 29533 | R | E | 13488 |
| 29534 | R | E | 13488 |
| 29535 | R | E | 13488 |
| 29536 | R | E | 13488 |
| 29537 | R | E | 13488 |
| 29540 | R,E | E | 13488 |
| 29541 | R | E | 13488 |
| 29546 | E | E | 13488 |
| 29623 | R,E | E | 13488 |
| 29712 | R | E,C | 13488 |
| 29713 | R,M | E,C,M | 13488 |
| 29714 | R | E,C | 13488 |
| 29715 | R | E,C | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|-------------------|--|---|---------------|
| 29760 | R,M | E,C,M | 13488 |
| 32805 | R | E | 13488 |
| 33058 | R,C | E,C | 13488 |
| 33205 | R | E | 13488 |
| 33268 | R | E | 13488 |
| 33618 | R | E | 13488 |
| 33619 | R | E | 13488 |
| 33620 | R | E | 13488 |
| 33621 | R | E | 13488 |
| 33623 | R | E | 13488 |
| 33624 | R | E | 13488 |
| 33632 | R | E | 13488 |
| 33636 | R,E | E | 13488 |
| 33637 | R | E | 13488 |
| 33665 | R,M | E,C,M | 13488 |
| 37301 | R | E | 13488 |
| 37719 | R | E | 13488 |
| 37728 | R | E | 13488 |
| 37732 | R,E | E | 13488 |
| 37761 | R,M | E,C,M | 13488 |
| 41397 | R | E | 13488 |
| 41460 | R | E | 13488 |
| 41824 | R | E | 13488 |
| 41828 | R,E | E | 13488 |
| 45493 | R | E | 13488 |
| 45556 | R | E | 13488 |
| 45920 | R | E | 13488 |
| 49589 | R | E | 13488 |
| 49652 | R | E | 13488 |
| 53668 | R,L | E,L | 13488 |
| 53685 | R | E | 17584 |
| 53748 | R | E | 13488 |
| 54189 | R | E | 13488 |
| 54289 | R | E | 13488 |

Table 62. Direct Conversions Supported to and from Unicode CCSID 01200 (continued)

| Non-Unicode CCSID | Techniques supported converting to Unicode | Technique supported converting from Unicode | Unicode CCSID |
|--------------------------|---|--|----------------------|
| 61696 | R | E | 13488 |
| 61697 | R | E | 13488 |
| 61698 | R | E | 13488 |
| 61699 | R | E | 13488 |
| 61700 | R | E | 13488 |
| 61710 | R | E | 13488 |
| 61711 | R | E | 13488 |
| 61712 | R | E | 13488 |
| 62273 | R,L | E,L | 21680 |
| 62337 | R | E | 13488 |
| 62381 | R | E | 13488 |

Appendix D. Validation, case, normalization, collation, & stringprep resources

The following conversion tables are supplied:

- Validation tables
- Case conversion tables
- Normalization tables
- Collation tables
- Stringprep tables

Validation tables

The following table lists the support provided by IBM for use on the character conversion service to support validation. See the CUNBCPRM_Mal_Action parameter for more detail.

| Table 63. Character conversion service supporting validation | |
|--|--------|
| Input CCSID | Table |
| 300 | CUNVBQ |
| 301 | CUNVBV |
| 367 | CUNVB0 |
| 834 | CUNVDM |
| 835 | CUNVDR |
| 837 | CUNVDY |
| 926 | CUNVIH |
| 927 | CUNVIJ |
| 928 | CUNVIM |
| 941 | CUNVJP |
| 947 | CUNVJ9 |
| 951 | CUNVKS |
| 1200 | CUNVPF |
| 1351 | CUNVQI |
| 1362 | CUNVQJ |
| 1374 | CUNVTZ |
| 1380 | CUNVQV |
| 1382 | CUNVQ0 |
| 1385 | CUNVQ6 |
| 4390 | CUNVDN |
| 4396 | CUNVBR |

| <i>Table 63. Character conversion service supporting validation (continued)</i> | |
|---|--------------|
| Input CCSID | Table |
| 4933 | CUNVDZ |
| 5043 | CUNVKA |
| 5047 | CUNVKT |
| 5470 | CUNVT2 |
| 5478 | CUNVQ1 |
| 9026 | CUNVDO |
| 9027 | CUNVDT |
| 9139 | CUNVKB |
| 12588 | CUNVBT |
| 13125 | CUNVTJ |
| 13235 | CUNVKC |
| 13488 | CUNVPG |
| 16684 | CUNVBU |
| 17221 | CUNVTL |
| 17584 | CUNVPH |
| 20780 | CUNVTQ |
| 21427 | CUNVKE |
| 21680 | CUNVTH |
| 42160 | CUNVUR |
| 54448 | CUNVVA |

Case conversion tables

These tables are provided by IBM for case conversion service.

| <i>Table 64. Case conversion service based on the Unicode Standard 3.0.1.</i> | | |
|---|---------------------|-------------|
| Table name | Description | Size |
| CUNA301C | to Upper Normal | 128K |
| 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 | Title stops table | 128K |
| CUNA301J | To Title | 128K |
| CUNA301K | INITCAP stops table | 128K |

Table 64. Case conversion service based on the Unicode Standard 3.0.1. (continued)

| Table name | Description | Size |
|------------|---------------------|------|
| CUNA301Y | Special Casing file | 32K |

Table 65. 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 |
| CUNA320K | INITCAP stops table | 128K |
| CUNA320S | to Upper Normal Surrogates | 0.5K |
| CUNA320T | to lower Normal Surrogates | 0.5K |
| CUNA320Y | Special Casing file | 32K |

Table 66. 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 |
| CUNA401K | INITCAP stops table | 128K |
| CUNA401S | to Upper Normal Surrogates | 0.5K |
| CUNA401T | to lower Normal Surrogates | 0.5K |
| CUNA401Y | Special Casing file | 32K |

Table 67. Case conversion service based on the Unicode Standard 4.1.0.

| Table name | Description | Size |
|------------|-----------------|------|
| CUNA410C | to Upper Normal | 128K |
| CUNA410D | to Lower Normal | 128K |

Table 67. Case conversion service based on the Unicode Standard 4.1.0. (continued)

| Table name | Description | Size |
|------------|----------------------------|------|
| 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 |
| CUNA410K | INITCAP stops table | 128K |
| CUNA410S | to Upper Normal Surrogates | 0.5K |
| CUNA410T | to lower Normal Surrogates | 0.5K |
| CUNA410Y | Special Casing file | 32K |

Table 68. 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 |
| CUNA500K | INITCAP stops table | 128K |
| CUNA500S | to Upper Normal Surrogates | 0.5K |
| CUNA500T | to lower Normal Surrogates | 0.5K |
| CUNA500Y | Special Casing file | 32K |

Table 69. Case conversion service based on the Unicode Standard 6.0.0.

| Table name | Description | Size |
|------------|--------------------|------|
| CUNA600C | to Upper Normal | 131K |
| CUNA600D | to Lower Normal | 131K |
| CUNA600E | to Upper Special | 131K |
| CUNA600F | to lower Special | 131K |
| CUNA600G | to Upper Locale | 131K |
| CUNA600H | to lower Locale | 131K |
| CUNA600I | Tittle stops table | 131K |

Table 69. Case conversion service based on the Unicode Standard 6.0.0. (continued)

| Table name | Description | Size |
|------------|----------------------------|-------|
| CUNA600J | To Title | 131K |
| CUNA600K | INITCAP stops table | 131K |
| CUNA600S | to Upper Normal Surrogates | 0.5K |
| CUNA600T | to lower Normal Surrogates | 0.5K |
| CUNA600Y | Special Casing file | 35.5K |

Table 70. Case conversion service based on the Unicode Standard 9.0.0.

| Table name | Description | Size |
|------------|----------------------------|------|
| CUNA900C | to Upper Normal | 128K |
| CUNA900D | to Lower Normal | 128K |
| CUNA900E | to Upper Special | 128K |
| CUNA900F | to lower Special | 128K |
| CUNA900G | to Upper Locale | 128K |
| CUNA900H | to lower Locale | 128K |
| CUNA900I | Tittle stops table | 128K |
| CUNA900J | To Title | 128K |
| CUNA900K | INITCAP stops table | 128K |
| CUNA900S | to Upper Normal Surrogates | 0.5K |
| CUNA900T | to lower Normal Surrogates | 0.5K |
| CUNA900Y | Special Casing file | 32K |

Table 71. Case conversion service based on the Unicode Standard 13.0.0.

| Table name | Description | Size |
|------------|----------------------------|------|
| CUNAX13C | to Upper Normal | 128K |
| CUNAX13D | to Lower Normal | 128K |
| CUNAX13E | to Upper Special | 128K |
| CUNAX13F | to lower Special | 128K |
| CUNAX13G | to Upper Locale | 128K |
| CUNAX13H | to lower Locale | 128K |
| CUNAX13I | Tittle stops table | 128K |
| CUNAX13J | To Title | 128K |
| CUNAX13K | INITCAP stops table | 128K |
| CUNAX13S | to Upper Normal Surrogates | 0.5K |
| CUNAX13T | to lower Normal Surrogates | 0.5K |
| CUNAX13Y | Special Casing file | 32K |

Normalization tables

These tables are provided by IBM for normalization service.

| <i>Table 72. Normalization service based on the Unicode Standard 3.0.1.</i> | | |
|---|-----------------------------------|--------|
| 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 |
| CUNCCNZ | Canonical class non zero | 64K |

| <i>Table 73. Normalization service based on the Unicode Standard 3.2.0.</i> | | |
|---|---|--------|
| 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 |

| <i>Table 74. Normalization service based on the Unicode Standard 4.0.1.</i> | | |
|---|-------------------------------|--------|
| Table name | Description | Size |
| CUNN401A | Canonical Decomposition Table | 10.25K |

Table 74. Normalization service based on the Unicode Standard 4.0.1. (continued)

| Table name | Description | Size |
|------------|---|--------|
| 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 |
| 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 75. 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 |

Table 75. Normalization service based on the Unicode Standard 4.1.0. (continued)

| Table name | Description | Size |
|-------------------|--|-------------|
| CUNN410L | Canonical Class Non Zero for supplementary code points | 0.025K |

Table 76. Normalization service based on the Unicode Standard 6.0.0.

| Table name | Description | Size |
|-------------------|---|-------------|
| CUNN600A | Canonical Decomposition Table | 11.75K |
| CUNN600B | Canonical Decomposition Stop Table | 128K |
| CUNN600C | Compatibility Decomposition Table | 37.25K |
| CUNN600D | Compatibility Decomposition Stop Table | 128K |
| CUNN600E | Composition Table | 7.5K |
| CUNN600F | Composition Stop Table | 128K |
| CUNN600G | Canonical Class Table | 64K |
| CUNN600H | Canonical Class Non Zero | 64K |
| CUNN600I | Canonical Decomposition Table for supplementary code points | 8K |
| CUNN600J | Compatibility Decomposition Table for supplementary code points | 52.25K |
| CUNN600K | Composition Table for supplementary code points | 8K |
| CUNN600L | Canonical Class Non Zero for supplementary code points | 0.5K |

Table 77. Normalization service based on the Unicode Standard 9.0.0.

| Table name | Description | Size |
|-------------------|--|-------------|
| CUNN900A | Canonical Decomposition Table | 11.75K |
| CUNN900B | Canonical Decomposition Stop Table | 128K |
| CUNN900C | Compatibility Decomposition Table | 37.25K |
| CUNN900D | Compatibility Decomposition Stop Table | 128K |
| CUNN900E | Composition Table | 7.5K |
| CUNN900F | Composition Stop Table | 128K |
| CUNN900G | Canonical Class Table | 64K |
| CUNN900H | Canonical Class Non Zero | 64K |

Table 77. Normalization service based on the Unicode Standard 9.0.0. (continued)

| Table name | Description | Size |
|------------|---|--------|
| CUNN900I | Canonical Decomposition Table for supplementary code points | 9K |
| CUNN900J | Compatibility Decomposition Table for supplementary code points | 56.75K |
| CUNN900K | Composition Table for supplementary code points | 8.75K |
| CUNN900L | Canonical Class Non Zero for supplementary code points | 1K |

Table 78. Normalization service based on the Unicode Standard 13.0.0.

| Table name | Description | Size |
|------------|---|--------|
| CUNNX13A | Canonical Decomposition Table | 11.75K |
| CUNNX13B | Canonical Decomposition Stop Table | 128K |
| CUNNX13C | Compatibility Decomposition Table | 37.25K |
| CUNNX13D | Compatibility Decomposition Stop Table | 128K |
| CUNNX13E | Composition Table | 7.5K |
| CUNNX13F | Composition Stop Table | 128K |
| CUNNX13G | Canonical Class Table | 64K |
| CUNNX13H | Canonical Class Non Zero | 64K |
| CUNNX13I | Canonical Decomposition Table for supplementary code points | 9K |
| CUNNX13J | Compatibility Decomposition Table for supplementary code points | 56.75K |
| CUNNX13K | Composition Table for supplementary code points | 8.75K |
| CUNNX13L | Canonical Class Non Zero for supplementary code points | 1K |

Collation tables

These tables are provided by IBM for collation service.

Table 79. Collation service based on the Unicode Standard 3.0.1.

| Table name | Description | Size |
|------------|--------------------------------|------|
| CUNOBACE | Collation element (main) table | 256K |
| CUNOMIDX | Index table | 64K |

Table 79. Collation service based on the Unicode Standard 3.0.1. (continued)

| Table name | Description | Size |
|------------|---------------------------------------|--------|
| CUNOTHLA | Thai Lao table | 64K |
| CUNOFCD | Fast canonical decomposition stop | 64K |
| 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 80. 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 81. 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 |
| 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 |

Table 82. Collation service based on the Unicode Standard 6.0.0.

| Table name | Description | Size |
|-------------------|---|-------------|
| CUNO600A | Collation Element Main Table | 640K |
| CUNO600B | Expansion Index Table | 192K |
| CUNO600C | Expansion Elements Table | 522.5K |
| CUNO600D | Contractions Index Table | 32.25K |
| CUNO600E | Contractions Elements Table | 9.25K |
| CUNO600F | Main Index Table | 64K |
| CUNO600G | Rearrangement Values | 64K |
| CUNO600H | Fast Canonical Decomposition | 64K |
| CUNO600I | Fast Compatibility Decomposition | 64K |
| CUNO600J | Fast Composition | 64K |
| CUNO600K | Surrogates Collation Element Main Table | 685.5K |

Table 82. Collation service based on the Unicode Standard 6.0.0. (continued)

| Table name | Description | Size |
|------------|---|--------|
| CUNO600L | Surrogates Expansion Elements Table | 23.25K |
| CUNO600M | Surrogates Contractions Elements Table | 0.25K |
| CUNO600N | Surrogates Main Index Table | 750.5K |
| CUNO600O | Surrogates Fast Canonical Decomposition | 1.75K |
| CUNO600P | Surrogates Fast Compatibility Decomposition | 5K |
| CUNO600Q | Surrogates Fast Composition | 0.25K |
| CUNA600A | Upper Case Attribute Table | 131K |
| CUNA600B | Lower Case Attribute Table | 131K |
| CUNA600L | Upper Case Attribute Table | 0.25K |
| CUNA600M | Lower Case Attribute Table | 0.25K |

Table 83. Collation service based on the Unicode Standard 9.0.0.

| Table name | Description | Size |
|------------|---|---------|
| CUNO900A | Collation Element Main Table | 640K |
| CUNO900B | Expansion Index Table | 192K |
| CUNO900C | Expansion Elements Table | 514.5K |
| CUNO900D | Contractions Index Table | 33.25K |
| CUNO900E | Contractions Elements Table | 11K |
| CUNO900F | Main Index Table | 64K |
| CUNO900G | Rearrangement Values | 64K |
| CUNO900H | Fast Canonical Decomposition | 64K |
| CUNO900I | Fast Compatibility Decomposition | 64K |
| CUNO900J | Fast Composition | 64K |
| CUNO900K | Surrogates Collation Element Main Table | 837K |
| CUNO900L | Surrogates Expansion Elements Table | 21.25K |
| CUNO900M | Surrogates Contractions Elements Table | 0.25K |
| CUNO900N | Surrogates Main Index Table | 912.25K |
| CUNO900O | Surrogates Fast Canonical Decomposition | 1.75K |
| CUNO900P | Surrogates Fast Compatibility Decomposition | 5.5K |

Table 83. Collation service based on the Unicode Standard 9.0.0. (continued)

| Table name | Description | Size |
|------------|-----------------------------|-------|
| CUNO900Q | Surrogates Fast Composition | 0.25K |
| CUNA900A | Upper Case Attribute Table | 128K |
| CUNA900B | Lower Case Attribute Table | 128K |
| CUNA900L | Upper Case Attribute Table | 1.25K |
| CUNA900M | Lower Case Attribute Table | 1.25K |

Table 84. Collation service based on the Unicode Standard 13.0.0.

| Table name | Description | Size |
|------------|---|---------|
| CUNOX13A | Collation Element Main Table | 640K |
| CUNOX13B | Expansion Index Table | 192K |
| CUNOX13C | Expansion Elements Table | 514.5K |
| CUNOX13D | Contractions Index Table | 33.25K |
| CUNOX13E | Contractions Elements Table | 11K |
| CUNOX13F | Main Index Table | 64K |
| CUNOX13G | Rearrangement Values | 64K |
| CUNOX13H | Fast Canonical Decomposition | 64K |
| CUNOX13I | Fast Compatibility Decomposition | 64K |
| CUNOX13J | Fast Composition | 64K |
| CUNOX13K | Surrogates Collation Element Main Table | 837K |
| CUNOX13L | Surrogates Expansion Elements Table | 21.25K |
| CUNOX13M | Surrogates Contractions Elements Table | 0.25K |
| CUNOX13N | Surrogates Main Index Table | 912.25K |
| CUNOX13O | Surrogates Fast Canonical Decomposition | 1.75K |
| CUNOX13P | Surrogates Fast Compatibility Decomposition | 5.5K |
| CUNOX13Q | Surrogates Fast Composition | 0.25K |
| CUNAX13A | Upper Case Attribute Table | 128K |
| CUNAX13B | Lower Case Attribute Table | 128K |
| CUNAX13L | Upper Case Attribute Table | 1.25K |
| CUNAX13M | Lower Case Attribute Table | 1.25K |

Stringprep tables

These profiles are provided by IBM for stringprep service.

| <i>Table 85. Profiles provided for stringprep service</i> | | |
|---|--|-------------|
| Profile name | Description | Size |
| CUNSTCIS | For Unix like filenames that are upper case only names | 64K |
| CUNSTCSP | For Unix like path and filenames | 8K |
| CUNSTMX1 | For (B.1) user name in name@domain | 8.5K |
| CUNSTMX2 | For B.1+B.2 domain name in name@domain | 64K |

Appendix E. Locales for collation and case support

Locales supported for collation

Table 86 on page 537 lists the locales supported in the data set SYS1.SCUNLOCL.

Note: Not all locales are supported for all collation versions. Each locale member name listed in Table 86 on page 537 has a prefix of either CUN or CUO, which is based on the collation version:

| Collation version | Locale member name prefix |
|-------------------|---------------------------|
| UCA400R1 | CUN |
| UCA410 | CUN |
| UCA600 | CUO |
| UCA900 | CUO |
| UCAX13 | CUO |

If the requested locale (for example, CUNBOPRM_Locale or CUN4BOPR_Locale) is not available for the specified collation version (CUNBOPRM_UCA_VER or CUN4BOPR_UCA_VER), an error is returned.

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) | | | | | |
|---|-----------------|------------------------------|----------------------|---------|--|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 1 | LAF | Afrikaans | | | <ul style="list-style-type: none"> CUNAF CUOAF |
| 2 | LAF_RNA | Afrikaans | Namibia Africa | | CUOAFNA |
| 3 | LAF_RZA | Afrikaans | South Africa | | <ul style="list-style-type: none"> CUNAFZA CUOAFZA |
| 4 | LAM | Amharic | | | CUNAM |
| 5 | LAM_RET | Amharic | Ethiopia | | CUNAMET |
| 6 | LAR | Arabic | | | <ul style="list-style-type: none"> CUNAR CUOAR |
| 7 | LAR_RAE | Arabic | United Arab Emirates | | <ul style="list-style-type: none"> CUNARAE CUOARAE |
| 8 | LAR_RBH | Arabic | Bahrain | | <ul style="list-style-type: none"> CUNARBH CUOARBH |
| 9 | LAR_RDZ | Arabic | Algeria | | <ul style="list-style-type: none"> CUNARDZ CUOARDZ |
| 10 | LAR_REG | Arabic | Egypt | | <ul style="list-style-type: none"> CUNAREG CUOAREG |
| 11 | LAR_RIN | Arabic | India | | CUNARIN |
| 12 | LAR_RIQ | Arabic | Iraq | | <ul style="list-style-type: none"> CUNARIQ CUOARIQ |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|-----------------|------------------------------|--------------|---------|--|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 13 | LAR_RJO | Arabic | Jordan | | <ul style="list-style-type: none"> CUNARJO CUOARJO |
| 14 | LAR_RKW | Arabic | Kuwait | | <ul style="list-style-type: none"> CUNARKW CUOARKW |
| 15 | LAR_RLB | Arabic | Lebanon | | <ul style="list-style-type: none"> CUNARLB CUOARLB |
| 16 | LAR_RLY | Arabic | Libya | | <ul style="list-style-type: none"> CUNARLY CUOARLY |
| 17 | LAR_RMA | Arabic | Morocco | | <ul style="list-style-type: none"> CUNARMA CUOARMA |
| 18 | LAR_ROM | Arabic | Oman | | <ul style="list-style-type: none"> CUNAROM CUOAROM |
| 19 | LAR_RQA | Arabic | Qatar | | <ul style="list-style-type: none"> CUNARQA CUOARQA |
| 20 | LAR_RSA | Arabic | Saudi Arabia | | <ul style="list-style-type: none"> CUNARSA CUOARSA |
| 21 | LAR_RSD | Arabic | Sudan | | <ul style="list-style-type: none"> CUNARSD CUOARSD |
| 22 | LAR_RSY | Arabic | Syria | | <ul style="list-style-type: none"> CUNARSY CUOARSY |
| 23 | LAR_RTN | Arabic | Tunisia | | <ul style="list-style-type: none"> CUNARTN CUOARTN |
| 24 | LAR_RYE | Arabic | Yemen | | <ul style="list-style-type: none"> CUNARYE CUOARYE |
| 25 | LAS | Assamese | | | CUOAS |
| 26 | LAS_RIN | Assamese | India | | CUOASIN |
| 27 | LAZ | Azeri | | | CUOAZ |
| 28 | LAZ_RAZ | Azeri | Azerbaijan | | CUOAZAZ |
| 29 | LAZ_VE | Azeri | | Search | CUOAZE |
| 30 | LBE | Belarusian | | | <ul style="list-style-type: none"> CUNBE CUOBE |
| 31 | LBE_RBY | Belarusian | Belarus | | <ul style="list-style-type: none"> CUNBEBY CUOEBY |
| 32 | LBG | Belarusian | | | <ul style="list-style-type: none"> CUNBG CUOBG |
| 33 | LBG_RBG | Belarusian | Bulgaria | | <ul style="list-style-type: none"> CUNBGBG CUOBGBG |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|------------------|------------------------------|------------------------|------------------|--|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 34 | LBN | Bengali | | | <ul style="list-style-type: none"> • CUNBN • CUOBN |
| 35 | LBN_RBD | Bengali | Bangladesh | | CUOBNBD |
| 36 | LBN_RIN | Bengali | India | | <ul style="list-style-type: none"> • CUNBNIN • CUOBNIN |
| 37 | LBN_VTRADITIONAL | Bengali | | Traditional | CUOBNNT |
| 38 | LBS | Bosnian | | | CUOBS |
| 39 | LBS_RBA | Bosnian | Bosnia and Herzegovina | | CUOBSBA |
| 40 | LCA | Catalan | | | <ul style="list-style-type: none"> • CUNCA • CUOCA |
| 41 | LCA_VSEARCH | Catalan | | Search | CUOCAE |
| 42 | LCA_RES | Catalan | Spain | | <ul style="list-style-type: none"> • CUNCAES • CUOCAES |
| 43 | LCA_RES_VPREEURO | Catalan | Spain | Pre Euro support | CUNCAESP |
| 44 | LCS | Czech | | | <ul style="list-style-type: none"> • CUNCSCS • CUOCS |
| 45 | LCS_RCZ | Czech | Czech Republic | | <ul style="list-style-type: none"> • CUNCSCSZ • CUOCSZ |
| 46 | LCY | Welsh | | | CUOCY |
| 47 | LCY_RGB | Welsh | United Kingdom | | CUOCYGB |
| 48 | LDA | Danish | | | <ul style="list-style-type: none"> • CUNDA • CUODA |
| 49 | LDA_RDK | Danish | Denmark | | <ul style="list-style-type: none"> • CUNDADK • CUODADK |
| 50 | LDA_VSEARCH | Danish | | Search | CUODAE |
| 51 | LDE | German | | | <ul style="list-style-type: none"> • CUNDE • CUODE |
| 52 | LDE_RAT | German | Austria | | <ul style="list-style-type: none"> • CUNDEAT • CUODEAT |
| 53 | LDE_RAT_VPREEURO | German | Austria | Pre Euro support | CUNDEATP |
| 54 | LDE_RBE | German | Belgin | | <ul style="list-style-type: none"> • CUNDEBE • CUODEBE |
| 55 | LDE_RCH | German | Switzerland | | <ul style="list-style-type: none"> • CUNDECH • CUODECH |
| 56 | LDE_RDE | German | Germany | | <ul style="list-style-type: none"> • CUNDEDE • CUODEDE |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|------------------|------------------------------|--------------------------|------------------|--|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 57 | LDE_RDE_PREEURO | German | Germany | Pre Euro support | CUNDEDEP |
| 58 | LDE_VSEARCH | German | | Search | CUODEE |
| 59 | LDE_VPHONEBOOK | German | | Telephone book | <ul style="list-style-type: none"> • CUNDEH • CUODEH |
| 60 | LDE_RLI | German | Liechtenstein | | CUODELI |
| 61 | LDE_RLU | German | Luxembourg | | <ul style="list-style-type: none"> • CUNDELU • CUODELU |
| 62 | LDE_RLU_PREEURO | German | Luxembourg | Pre Euro support | CUNDELUP |
| 63 | LEL | Greek | | | <ul style="list-style-type: none"> • CUNEL • CUOEL |
| 64 | LEL_RCY | Greek | Cyprus | | CUOELCY |
| 65 | LEL_RGR | Greek | Greece | | <ul style="list-style-type: none"> • CUNELGR • CUOELGR |
| 66 | LEL_RGR_VPREEURO | Greek | Greece | Pre Euro support | CUNELGRP |
| 67 | LEN | English | | | <ul style="list-style-type: none"> • CUNEN • CUOEN |
| 68 | LEN_RAS | English | American Samoa | | CUOENAS |
| 69 | LEN_RAU | English | Australia | | CUNENAU CUOENAU |
| 70 | LEN_RBE | English | Belgium | | CUNENBE CUOENBE |
| 71 | LEN_RBE_VPREEURO | English | Belgium | Pre Euro support | CUNENBEP |
| 72 | LEN_RBW | English | Botswana | | CUNENBW CUOENBW |
| 73 | LEN_RBZ | English | Belize | | CUOENBZ |
| 74 | LEN_RCA | English | Canada | | CUNENCA CUOENCA |
| 75 | LEN_RGB | English | Great Britain | | CUNENGB CUOENGB |
| 76 | LEN_RGB_VPREEURO | English | Great Britain | Pre Euro support | CUNENGBP |
| 77 | LEN_RGU | English | Guam | | CUOENGU |
| 78 | LEN_RHK | English | Hong Kong S.A.R of China | | CUNENHK CUOENHK |
| 79 | LEN_RIE | English | Ireland | | CUNENIE CUOENIE |

Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
|-----|------------------|------------------------------|-----------------------------|------------------|----------------------|
| 80 | LEN_RIE_VPREEURO | English | Ireland | Pre Euro support | CUNENIEP |
| 81 | LEN_RIN | English | India | | CUNENIN CUOENIN |
| 82 | LEN_RJM | English | Jamaica | | CUOENJM |
| 83 | LEN_RMH | English | Marshall Islands | | CUOENMH |
| 84 | LEN_RMP | English | Northern Mariana Islands | | CUOENMP |
| 85 | LEN_RMT | English | Malta | | CUNENMT CUOENMT |
| 86 | LEN_RMU | English | Mauritius | | CUOENMU |
| 87 | LEN_RNA | English | Namibia | | CUOENNA |
| 88 | LEN_RNZ | English | New Zealand | | CUNENNZ CUOENNZ |
| 89 | LEN_RPH | English | Philippines | | CUNENPH CUOENPH |
| 90 | LEN_RPK | English | Pakistan | | CUOENPK |
| 91 | LEN_RSG | English | Singapore | | CUNENSG CUOENSG |
| 92 | LEN_RTT | English | Trinidad | | CUOENTT |
| 93 | LEN_RUM | English | U.S. Minor Outlying Islands | | CUOENUM |
| 94 | LEN_RUS | English | United States of America | | CUNENUS CUOENUS |
| 95 | LEN_RUS_VPOSIX | English | United States of America | Posix | CUNENUSX CUOENUSX |
| 96 | LEN_RVI | English | Virgin Islands (USA) | | CUNENVI CUOENVI |
| 97 | LEN_RZA | English | South Africa | | CUNENZA CUOENZA |
| 98 | LEN_RZW | English | Zimbabwe | | CUNENZW CUOENZW |
| 99 | LEO | Esperanto | | | CUNEO CUOEO |
| 100 | LES | Spanish | | | CUNES CUOES |
| 102 | LES_RAR | Spanish | Argentina | | CUNESAR CUOESAR |
| 103 | LES_RBO | Spanish | Bolivia | | CUNESBO CUOESBO |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|------------------|------------------------------|--------------------------|--------------------------|----------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 104 | LES_RCL | Spanish | Chile | | CUNESCL CUOESCL |
| 105 | LES_RCO | Spanish | Colombia | | CUNESCO CUOESCO |
| 106 | LES_RCR | Spanish | Costa Rica | | CUNESCR CUOESCR |
| 107 | LES_RDO | Spanish | Dominican Republic | | CUNESDO CUOESDO |
| 108 | LES_VSEARCH | Spanish | | Search | CUOESE |
| 109 | LES_REC | Spanish | Ecuador | | CUNESEC CUOESSEC |
| 110 | LES_RES | Spanish | Spain | | CUNESSES CUOESSES |
| 111 | LES_RES_VPREEURO | Spanish | Spain | | CUNESESP |
| 112 | LES_RGQ | Spanish | Equatorial Guinea | | CUOESGQ |
| 113 | LES_RGT | Spanish | Guatemala | | CUNESGT CUOESGT |
| 114 | LES_RHN | Spanish | Honduras | | CUNESHN CUOESHN |
| 115 | LES_RMX | Spanish | Mexico | | CUNESMX CUOESMX |
| 116 | LES_RNI | Spanish | Nicaragua | | CUNESNI CUOESNI |
| 117 | LES_RPA | Spanish | Panama | | CUNESPA CUOESPA |
| 118 | LES_RPE | Spanish | Peru | | CUNESPE CUOESPE |
| 119 | LES_RPR | Spanish | Puerto Rico | | CUNESPR CUOESPR |
| 120 | LES_RPY | Spanish | Paraguay | | CUNESPY CUOESPY |
| 121 | LES_RSV | Spanish | El Salvador | | CUNESSV CUOESSV |
| 122 | LES_VTRADITIONAL | Spanish | | Traditional Spanish sort | CUNEST CUOEST |
| 123 | LES_RUS | Spanish | United States of America | | CUNESUS CUOESUS |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|------------------|------------------------------|---------------|------------------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 12 4 | LES_RUY | Spanish | Uruguay | | CUNESUY CUOESUY |
| 12 5 | LES_RVE | Spanish | Venezuela | | CUNESVE CUOESVE |
| 12 6 | LET | Estonian | | | CUNET CUOET |
| 12 7 | LET_REE | Estonian | Estonia | | CUNETEE CUOETEE |
| 12 8 | LEU | Basque | | | CUNEU |
| 12 9 | LEU_RES | Basque | Spain | | CUNEUES |
| 13 0 | LEU_RES_VPREEURO | Basque | Spain | Pre Euro support | CUNEUESP |
| 13 1 | LFA | Persian | | | CUNFA CUOFA |
| 13 2 | LFA_RAF | Persian | Afghanistan | | CUOFAAF |
| 13 3 | LFA_RIR | Persian | Iran | | CUNFAIR CUOFAIR |
| 13 4 | LFI | Finnish | | | CUNFI CUOFI |
| 13 5 | LFI_VSEARCH | Finnish | | Search | CUOFIE |
| 13 6 | LFI_RFI | Finnish | Finland | | CUNFIFI CUOFIFI |
| 13 7 | LFI_RFI_VPREEURO | Finnish | Finland | Pre Euro support | CUNFIFIP |
| 13 8 | LFI_VPHONEBOOK | Finnish | | Telephone book | CUOFIH |
| 13 9 | LFIL | Filipino | | | CUOFIL |
| 14 0 | LFIL_RPH | Filipino | Philippines | | CUOFILPH |
| 14 1 | LFO | Faroese | | | CUNFO CUOFO |
| 14 2 | LFO_VSEARCH | Faroese | | Search | CUOFOE |
| 14 3 | LFO_RFO | Faroese | Faroe Islands | | CUNFOFO CUOFOFO |
| 14 4 | LFR | French | | | CUNFR CUOFR |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|------------------|------------------------------|----------------------------------|------------------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 14 5 | LFR_RBE | French | Belgium | | CUNFRBE CUOFRBE |
| 14 6 | LFR_RBE_VPREEURO | French | Belgium | Pre Euro support | CUNFRBEP |
| 14 7 | LFR_RBF | French | Burkina | | CUOFRBF |
| 14 8 | LFR_RBI | French | Burundi | | CUOFRBI |
| 14 9 | LFR_RBJ | French | Benin | | CUOFRBJ |
| 15 0 | LFR_RBL | French | Saint Barthelemy | | CUOFRBL |
| 15 1 | LFR_RCA | French | Canada | | CUNFRCA CUOFRCA |
| 15 2 | LFR_RCD | French | Democratic Republic of the Congo | | CUOFRCD |
| 15 3 | LFR_RCF | French | Central African Republic | | CUOFRCF |
| 15 4 | LFR_RCG | French | Congo | | CUOFRCG |
| 15 5 | LFR_RCH | French | Switzerland | | CUNFRCH CUOFRCH |
| 15 6 | LFR_RCI | French | Cote d'Ivoire | | CUOFRCI |
| 15 7 | LFR_RCM | French | Cameroon | | CUOFRCM |
| 15 8 | LFR_RDJ | French | Djibouti | | CUOFRDJ |
| 15 9 | LFR_RFR | French | France | | CUNFRFR CUOFRFR |
| 16 0 | LFR_RFR_VPREEURO | French | France | Pre Euro support | CUNFRFRP |
| 16 1 | LFR_RGA | French | Gabon | | CUOFRGA |
| 16 2 | LFR_RGN | French | Guinea | | CUOFRGN |
| 16 3 | LFR_RGP | French | Guadeloupe | | CUOFRGP |
| 16 4 | LFR_RGQ | French | Equatorial | | CUOFRGQ |
| 16 5 | LFR_RKM | French | Comoros | | CUOFRKM |
| 16 6 | LFR_RLU | French | Luxembourg | | CUNFRLU CUOFRLU |
| 16 7 | LFR_RLU_VPREEURO | French | Luxembourg | | CUNFRLUP |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|------------------|------------------------------|--------------|------------------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 168 | LFR_RMC | French | Monaco | | CUOFRMC |
| 169 | LFR_RMF | French | Saint Martin | | CUOFRMF |
| 170 | LFR_RMG | French | Madagascar | | CUOFRMG |
| 171 | LFR_RML | French | Mali | | CUOFRML |
| 172 | LFR_RMQ | French | Martinique | | CUOFRMQ |
| 173 | LFR_RNE | French | Niger | | CUOFRNE |
| 174 | LFR_RRE | French | Reunion | | CUOFRRE |
| 175 | LFR_RRW | French | Rwanda | | CUOFRRW |
| 176 | LFR_RSN | French | Senegal | | CUOFRSN |
| 177 | LFR_RTD | French | Chad | | CUOFRTD |
| 178 | LFR_RTG | French | Togo | | CUOFRTG |
| 179 | LGA | Irish | | | CUNGA |
| 180 | LGA_RIE | Irish | Ireland | | CUNGAIE |
| 181 | LGA_RIE_VPREEURO | Irish | Ireland | Pre Euro support | CUNGAIEP |
| 182 | LGL | Galician | | | CUNGL |
| 183 | LGL_RES | Galician | Spain | | CUNGLES |
| 184 | LGL_RES_VPREEURO | Galician | Spain | Pre Euro support | CUNGLESP |
| 185 | LGU | Gujarati | | | CUNGU CUOGU |
| 186 | LGU_RIN | Gujarati | India | | CUNGUIN CUOGUIN |
| 187 | LGV | Manx | Gaelic | | CUNGV |
| 188 | LGV_RGB | Manx | Gaelic | Great Britain | CUNGVGB |
| 189 | LHA | Hausa | | | CUOHA |
| 190 | LHA_RGH | Hausa | Ghana | | CUOHAGH |
| 191 | LHA_RNE | Hausa | Niger | | CUOHANE |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|------------------|------------------------------|-----------|---------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 19 2 | LHA_RNG | Hausa | Nigeria | | CUOHANG |
| 19 3 | LHE | Hebrew | | | CUNHE CUOHE |
| 19 4 | LHE_RIL | Hebrew | Israel | | CUNHEIL CUOHEIL |
| 19 5 | LHI | Hindi | | | CUNHI CUOHI |
| 19 6 | LHI_VDIRECT | Hindi | | Direct | CUOHID |
| 19 7 | LHI_RIN | Hindi | India | | CUNHIIN CUOHIIN |
| 19 8 | LHR | Croatian | | | CUNHR CUOHR |
| 19 9 | LHR_VSEARCH | Croatian | | Search | CUOHRE |
| 20 0 | LHR_RHR | Croatian | Croatia | | CUNHRHR CUOHRHR |
| 20 1 | LHU | Hungarian | | | CUNHU CUOHU |
| 20 2 | LHU_RHU | Hungarian | Hungary | | CUNHUHU CUOHUHU |
| 20 3 | LHY | Armenian | | | CUNHY CUOHY |
| 20 4 | LHY_RAM | Armenian | Armenia | | CUNHYAM CUOHYAM |
| 20 5 | LHY_RAM_VREVISED | Armenian | Armenia | Revised | CUNHYAMR |
| 20 6 | LID | Indonesian | | | CUNID |
| 20 7 | LID_RID | Indonesian | Indonesia | | CUNIDID |
| 20 8 | LIG | Igbo | | | CUOIG |
| 20 9 | LIG_RNG | Igbo | Nigeria | | CUOIGNG |
| 21 0 | LIS | Icelandic | | | CUNIS CUOIS |
| 21 1 | LIS_VSEARCH | Icelandic | | Search | CUOISE |
| 21 2 | LIS_RIS | Icelandic | Iceland | | CUNISIS CUOISIS |

Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
|-----|------------------|------------------------------|-------------|------------------|--------------------|
| 213 | LIT | Italian | | | CUNIT |
| 214 | LIT_RCH | Italian | Switzerland | | CUNITCH |
| 215 | LIT_RIT | Italian | Italy | | CUNITIT |
| 216 | LIT_RIT_VPREEURO | Italian | Italy | Pre Euro support | CUNITITP |
| 217 | LIW | Hebrew | | | CUNIW |
| 218 | LIW_RIL | Hebrew | Israel | | CUNIWIL |
| 219 | LJA | Japanese | | | CUNJA CUOJA |
| 220 | LJA_RJP | Japanese | Japan | | CUNJAJP CUOJAJP |
| 221 | LJA_VUNIHAN | Japanese | | Unihan | CUOJAU |
| 222 | LKK | Kazakh | | | CUOKK |
| 223 | LKK_RKZ | Kazakh | Kazakhstan | | CUOKKKZ |
| 224 | LKL | Greenlandic | | | CUNKL CUOKL |
| 225 | LKL_VSEARCH | Greenlandic | | Search | CUOKLE |
| 226 | LKL_RGL | Greenlandic | Greenland | | CUNKLGL CUOKLGL |
| 227 | LKM | Khmer | | | CUNKM |
| 228 | LKM_RKH | Khmer | Cambodia | | CUOKMKH |
| 229 | LKN | Kannada | | | CUNKN CUOKN |
| 230 | LKN_RIN | Kannada | India | | CUNKNIN CUOKNIN |
| 231 | LKN_VTRADITIONAL | Kannada | | Traditional | CUOKNT |
| 232 | LKO | Korean | | | CUNKO CUOKO |
| 233 | LKO_VSEARCH | Korean | | Search | CUOKOE |
| 234 | LKO_RKR | Korean | Korea | | CUNKOKR CUOKOKR |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|-----------------|------------------------------|---------------|---------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 23 5 | LKO_VUNIHAN | Korean | | Unihan | CUOKOU |
| 23 6 | LKOK | Konkani | | | CUOKOK |
| 23 7 | LKOK_RIN | Konkani | India | | CUOKOKIN |
| 23 8 | LK1 | Konkani | | | CUNK1 |
| 23 9 | LK1_RIN | KonKani | India | | CUNK1IN |
| 24 0 | LKW | Cornish | | | CUNKW |
| 24 1 | LKW_RGB | Cornish | Great Britain | | CUNKWGB |
| 24 2 | LLT | Lithuanian | | | CUNLT CUOLT |
| 24 3 | LLT_RLT | Lithuanian | Lithuania | | CUNLTLT CUOLTLT |
| 24 4 | LLV | Latvian | | | CUNLV CUOLV |
| 24 5 | LLV_RLV | Latvian | Latvia | | CUNLVLV CUOLVLV |
| 24 6 | LMK | Macedonian | | | CUNMK CUOMK |
| 24 7 | LMK_RMK | Macedonian | Macedonia | | CUNMKMK CUOMMKK |
| 24 8 | LML | Malayalam | | | CUOML |
| 24 9 | LML_RIN | Malayalam | India | | CUOMLIN |
| 25 0 | LMR | Marathi | | | CUNMR CUOMR |
| 25 1 | LMR_RIN | Marathi | India | | CUNMRIN CUOMRIN |
| 25 2 | LMT | Maltese | | | CUNMT CUOMT |
| 25 3 | LMT_RMT | Maltese | Malta | | CUNMTMT CUOMTMT |
| 25 4 | LMY | Burmese | | | CUOMY |
| 25 5 | LMY_RMM | Burmese | Myanmar | | CUOMYMM |
| 25 6 | LNB | Norwegian Bokmal | | | CUNNB CUONB |

Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
|-----|-----------------|------------------------------|-----------------|------------------|---------------------|
| 257 | LNB_RNO | Norwegian Bokmal | Norway | | CUNNBNO CUONBNO |
| 258 | LNL | Dutch | | | CUNNL |
| 259 | LNL_RBE | Dutch | Belgium | | CUNNLBE |
| 260 | LNL_RBE_VPREURO | Dutch | Belgium | Pre Euro support | CUNNLBEP |
| 261 | LNL_RNL | Dutch | The Netherlands | | CUNNLNL |
| 262 | LNL_RNL_VPREURO | Dutch | The Netherlands | Pre Euro support | CUNNLNLP |
| 263 | LNN | Norwegian Nynorsk | | | CUNNN CUONN |
| 264 | LNN_VSEARCH | Norwegian Nynorsk | | Search | CUONNE |
| 265 | LNN_RNO | Norwegian Nynorsk | Norway | | CUNNNNO CUONNNNO |
| 266 | LNSO | Pedi | | | CUONSO |
| 267 | LNSO_RZA | Pedi | South Africa | | CUONSOZA |
| 268 | LOM | Oromo | | | CUNOM CUOOM |
| 269 | LOM_RET | Oromo | Ethiopia | | CUNOMET CUOOMET |
| 270 | LOM_RKE | Oromo | Kenya | | CUNOMKE CUOOMKE |
| 271 | LOR | Oriya | | | CUOOR |
| 272 | LOR_RIN | Oriya | India | | CUOORIN |
| 273 | LPA | Punjabi | | | CUOPA |
| 274 | LPA_RIN | Punjabi | India | | CUOPAIN |
| 275 | LPA_RPK | Punjabi | Pakistan | | CUOPAPK |
| 276 | LPL | Polish | | | CUNPL CUOPL |
| 277 | LPL_RPL | Polish | Poland | | CUNPLPL CUOPLPL |
| 278 | LPS | Pushto | | | CUOPS |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|-----------------|------------------------------|-------------|------------------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 27 9 | LPS_RAF | Pushto | Afghanistan | | CUOPSAF |
| 28 0 | LPT | Portuguese | | | CUNPT |
| 28 1 | LPT_RBR | Portuguese | Brazil | | CUNPTBR |
| 28 2 | LPT_RPT | Portuguese | Portugal | | CUNPTPT |
| 28 3 | LPT_RPT_VPREURO | Portuguese | Portugal | Pre Euro support | CUNPTPTP |
| 28 4 | LRO | Romanian | | | CUNRO CUORO |
| 28 5 | LRO_RMD | Romanian | Moldova | | CUOROMD |
| 28 6 | LRO_RRO | Romanian | Romania | | CUNRORO CUORORO |
| 28 7 | LRU | Russian | | | CUNRU CUORU |
| 28 8 | LRU_RMD | Russian | Moldova | | CUORUMD |
| 28 9 | LRU_RRU | Russian | Russia | | CUNRURU CUORURU |
| 29 0 | LRU_RUA | Russian | Ukraine | | CUNRUUA CUORUUA |
| 29 1 | LSE | Northern Sami | | | CUOSE |
| 29 2 | LSE_VSEARCH | Northern Sami | | Search | CUOSEE |
| 29 3 | LSE_RFI | Northern Sami | Finland | | CUOSEFI |
| 29 4 | LSE_RNO | Northern Sami | Norway | | CUOSEN0 |
| 29 5 | LSH | Serbo-Croatian | | | CUNSH |
| 29 6 | LSH_RYU | Serbo-Croatian | Yugoslavia | | CUNSHYU |
| 29 7 | LSI | Sinhala-Sinhalese | | | CUOSI |
| 29 8 | LSI_VDICTIONARY | Sinhala-Sinhalese | | Dictionary | CUOSIC |
| 29 9 | LSI_RLK | Sinhala-Sinhalese | Sri Lanka | | CUOSILK |
| 30 0 | LSK | Slovak | | | CUNSK CUOSK |
| 30 1 | LSK_RSK | Slovak | Slovakia | | CUNSKSK CUOSKSK |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|-----------------|------------------------------|------------|----------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 30 2 | LSL | Slovenian | | | CUNSL CUOSL |
| 30 3 | LSL_RSI | Slovenian | Slovenia | | CUNSLSI CUOSLSI |
| 30 4 | LSO | Somali | | | CUNSO |
| 30 5 | LSO_RDJ | Somali | Djibouti | | CUNSODJ |
| 30 6 | LSO_RET | Somali | Ethiopia | | CUNSOET |
| 30 7 | LSO_RKE | Somali | Kenya | | CUNSOKE |
| 30 8 | LSO_RSO | Somali | Somalia | | CUNSOSO |
| 30 9 | LSQ | Albanian | | | CUNSQ CUOSQ |
| 31 0 | LSQ_RAL | Albanian | Albania | | CUNSQAL CUOSQAL |
| 31 1 | LSR | Serbian | | Cyrillic | CUNSR CUOSR |
| 31 2 | LSR_RBA | Serbian | Bosnia | Cyrillic | CUOSRBA |
| 31 3 | LSR_RBA_VSEARCH | Serbian | Bosnia | Search | CUOSRBAE |
| 31 4 | LSR_RBA_VLATIN | Serbian | Bosnia | Latin | CUOSRBAL |
| 31 5 | LSR_RME | Serbian | Montenegro | Cyrillic | CUOSRME |
| 31 6 | LSR_RME_VSEARCH | Serbian | Montenegro | Search | CUOSRMEE |
| 31 7 | LSR_RME_VLATIN | Serbian | Montenegro | Latin | CUOSRMEL |
| 31 8 | LSR_RRS | Serbian | Serbia | Cyrillic | CUOSRRS |
| 31 9 | LSR_RRS_VSEARCH | Serbian | Serbia | Search | CUOSRRSE |
| 32 0 | LSR_RRS_VLATIN | Serbian | Serbia | Latin | CUOSRRSL |
| 32 2 | LSR_RYU | Serbian | Yugoslavia | | CUNSRYU |
| 32 3 | LSV | Swedish | | | CUNSV CUOSV |
| 32 4 | LSV_VSEARCH | Swedish | | Search | CUOSVE |
| 32 5 | LSV_VREFORMED | Swedish | | Reformed | CUOSVF |

| Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued) | | | | | |
|---|-----------------|------------------------------|--------------|---------|--------------------|
| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
| 326 | LSV_RFI | Swedish | Finland | | CUNSVFI CUOSVFI |
| 327 | LSV_RSE | Swedish | Sweden | | CUNSVSE CUOSVSE |
| 328 | LSW | Swahili | | | CUNSW |
| 329 | LSW_RKE | Swahili | Kenya | | CUNSWKE |
| 330 | LSW_RTZ | Swahili | Tanzania | | CUNSWTZ |
| 331 | LTA | Tamil | | | CUNTA CUOTA |
| 332 | LTA_RIN | Tamil | India | | CUNTAIN CUOTAIN |
| 333 | LTA_RLK | Tamil | Sri Lanka | | CUOTALK |
| 334 | LTE | Telugu | | | CUNTE CUOTE |
| 335 | LTE_RIN | Telugu | India | | CUNTEIN CUOTEIN |
| 336 | LTH | Thai | | | CUNTH CUOTH |
| 337 | LTH_RTH | Thai | Thailand | | CUNTHTH CUOTHTH |
| 338 | LTI | Tigrinya | | | CUNTI |
| 339 | LTI_RER | Tigrinya | Eritrea | | CUNTIER |
| 340 | LTI_RET | Tigrinya | Ethiopia | | CUNTIET |
| 341 | LTN | Setsuana | | | CUOTN |
| 342 | LTN_RZA | Setsuana | South Africa | | CUOTNZA |
| 343 | LTR | Turkish | | | CUNTR CUOTR |
| 344 | LTR_VSEARCH | Turkish | | Search | CUOTRE |
| 345 | LTR_RTR | Turkish | Turkey | | CUNTRTR CUOTRTR |
| 346 | LUK | Ukrainian | | | CUNUK CUOUK |

Table 86. Locales support for CUNBOPRM_Collation_Keyword/CUN4BOPR_Collation_Keyword (31/64-bit) (continued)

| # | Parameter Value | Language or Ordering Profile | Region | Variant | Member Name |
|-----|-----------------|------------------------------|--------------------------|------------------|--------------------|
| 347 | LUK_RUA | Ukrainian | Ukrania | | CUNUKUA CUOUKUA |
| 348 | LUR | Urdu | | | CUOUR |
| 349 | LUR_RIN | Urdu | India | | CUOURIN |
| 350 | LUR_RPK | Urdu | Pakistan | | CUOURPK |
| 351 | LVI | Vietnamese | | | CUNVI CUOVI |
| 352 | LVI_RVN | Vietnamese | Vietnam | | CUNVIVN CUOVIVN |
| 353 | LWO | Wolof | | | CUOWO |
| 354 | LWO_RSN | Wolof | Senegal | | CUOWOSN |
| 355 | LYO | Yoruba | | | CUOYO |
| 356 | LYO_RNG | Yoruba | Nigeria | | CUOYONG |
| 357 | LZH | Chinese | | | CUNZH CUOZH |
| 358 | LZH_VBIG5HAN | Chinese | | BIG5HAN | CUOZHB |
| 359 | LZH_RCN | Chinese | China | | CUNZHCN CUOZHCN |
| 360 | LZH_VGB2312 | Chinese | | GB2312 | CUOZHG |
| 361 | LZH_RHK | Chinese | Hong Kong S.A.R of China | | CUNZHHK CUOZHHK |
| 362 | LZH_RMO | Chinese | Macao S.A.R of China | | CUNZHMO CUOZHMO |
| 363 | LZH_VSTROKE | Chinese | | Stroke ordering | CUOZHS |
| 364 | LZH_RSG | Chinese | Singapore | | CUNZHSG CUOZHSG |
| 365 | LZH_RTW | Chinese | Taiwan | | CUNZHTW CUOZHTW |
| 366 | LZH_RTW_VSTROKE | Chinese | Taiwan | Stroke ordering | CUNZHTWS |
| 367 | LZH_VUNIHAN | Chinese | | Unihan | CUOZHU |
| 368 | LZH_VPINYIN | Chinese | | Pin yin ordering | CUNZHY CUOZHY |

Locales supported for case service

This topic 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).

| <i>Table 87. Case service and locale valid names</i> | | |
|--|-----------------|--|
| Locale name | Language | 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 |

Table 87. Case service and locale valid names (continued)

| Locale name | Language | Region |
|-------------|------------|--------------------|
| Es_CR | Spanish | Costa Rica |
| Es_DO | Spanish | Dominican Republic |
| Es_EC | Spanish | Ecuador |
| 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 |

| <i>Table 87. Case service and locale valid names (continued)</i> | | |
|--|---------------------|-----------------|
| Locale name | Language | Region |
| 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 |
| 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 F. Locales for dynamic locale service

z/OS Unicode Services has a locale source repository that contains a copy of the Unicode common locale data repository locales (CLDR). This repository is used for the dynamic locale service. This repository can be found in the SYS1.SCUNTB1 data set.

Note: This repository also contains a character map, 01200.charmap, containing the symbolic names used in all of the locales provided in SYS1.SCUNTB1, along with their respective character mappings in Unicode.

Table 88. z/OS Unicode Services locale standard source repository

| Locale name | Member | Default CCSID (if defined) |
|-------------|----------|----------------------------|
| aa_DJ | CUOHL001 | 01200 |
| aa_ER | CUOHL002 | 01200 |
| aa_ET | CUOHL003 | 01200 |
| af_NA | CUOHL004 | 01200 |
| af_ZA | CUOHL005 | 01208 |
| ak_GH | CUOHL006 | 01200 |
| am_ET | CUOHL007 | 01200 |
| ar_AE | CUOHL009 | 01200 |
| ar_BH | CUOHL00A | 01200 |
| ar_DZ | CUOHL00B | 01200 |
| ar_EG | CUOHL00C | 01200 |
| ar_IQ | CUOHL00D | 01200 |
| ar_JO | CUOHL00E | 01200 |
| ar_KW | CUOHL00F | 01200 |
| ar_LB | CUOHL010 | 01200 |
| ar_LY | CUOHL011 | 01200 |
| ar_MA | CUOHL012 | 01200 |
| ar_OM | CUOHL013 | 01200 |
| ar_QA | CUOHL014 | 01200 |
| ar_SA | CUOHL015 | 01200 |
| ar_SD | CUOHL016 | 01200 |
| ar_SY | CUOHL017 | 01200 |
| ar_TN | CUOHL018 | 01200 |
| ar_YE | CUOHL019 | 01200 |
| as_IN | CUOHL01A | 01200 |
| asa_TZ | CUOHL01B | 01200 |
| az_Arab_IR | CUOHL01C | 01200 |

| <i>Table 88. z/OS Unicode Services locale standard source repository (continued)</i> | | |
|--|---------------|-----------------------------------|
| Locale name | Member | Default CCSID (if defined) |
| az_AZ | CUOHL01E | 01208 |
| az_Cyrl_AZ | CUOHL01D | 01200 |
| az_Latn_AZ | CUOHL01E | 01208 |
| be_BY | CUOHL01F | 01025 |
| bem_ZM | CUOHL020 | 01200 |
| bez_TZ | CUOHL021 | 01200 |
| bg_BG | CUOHL022 | 01025 |
| bg_BG@euro | CUOHL023 | 01025 |
| bg_BG@preeuro | CUOHL022 | 01025 |
| bm_ML | CUOHL024 | 01200 |
| bn_BD | CUOHL025 | 01200 |
| bn_IN | CUOHL026 | 01208 |
| bo_CN | CUOHL027 | 01200 |
| bo_IN | CUOHL028 | 01200 |
| br_FR | CUOHL029 | 01200 |
| brx_IN | CUOHL02A | 01200 |
| bs_BA | CUOHL02B | 01200 |
| byn_ER | CUOHL02C | 01200 |
| ca_ES | CUOHL02E | 00924 |
| ca_ES@euro | CUOHL02E | 00924 |
| ca_ES@preeuro | CUOHL02F | 00924 |
| cch_NG | CUOHL030 | 01200 |
| cgg_UG | CUOHL031 | 01200 |
| chr_US | CUOHL032 | 01200 |
| cs_CZ | CUOHL033 | 00870 |
| cs_CZ@euro | CUOHL034 | 00870 |
| cs_CZ@preeuro | CUOHL033 | 00870 |
| cy_GB | CUOHL035 | 01208 |
| cy_GB@euro | CUOHL036 | 01208 |
| da_DK | CUOHL037 | 01047 |
| da_DK@euro | CUOHL038 | 01047 |
| dav_KE | CUOHL039 | 01200 |
| de_AT | CUOHL03A | 00924 |
| de_AT@euro | CUOHL03A | 00924 |

Table 88. z/OS Unicode Services locale standard source repository (continued)

| Locale name | Member | Default CCSID (if defined) |
|---------------|----------|----------------------------|
| de_AT@preeuro | CUOHL03B | 00924 |
| de_BE | CUOHL03C | 01200 |
| de_CH | CUOHL03D | 01047 |
| de_CH@euro | CUOHL03E | 01047 |
| de_DE | CUOHL03F | 01047 |
| de_DE@euro | CUOHL03F | 01047 |
| de_DE@preeuro | CUOHL040 | 01047 |
| de_LI | CUOHL041 | 01200 |
| de_LU | CUOHL042 | 00924 |
| de_LU@euro | CUOHL042 | 00924 |
| de_LU@preeuro | CUOHL043 | 00924 |
| dv_MV | CUOHL044 | 01200 |
| dz_BT | CUOHL045 | 01200 |
| ebu_KE | CUOHL046 | 01200 |
| ee_GH | CUOHL047 | 01200 |
| ee_TG | CUOHL048 | 01200 |
| el_CY | CUOHL049 | 01200 |
| el_GR | CUOHL04A | 00875 |
| el_GR@euro | CUOHL04A | 00875 |
| el_GR@preeuro | CUOHL04B | 00875 |
| en_AS | CUOHL04C | 01200 |
| en_AU | CUOHL04D | 01047 |
| en_BE | CUOHL04E | 00924 |
| en_BE@euro | CUOHL04E | 00924 |
| en_BE@preeuro | CUOHL04F | 00924 |
| en_BW | CUOHL050 | 01200 |
| en_BZ | CUOHL051 | 01200 |
| en_CA | CUOHL052 | 01047 |
| en_GB | CUOHL053 | 01047 |
| en_GB@euro | CUOHL054 | 01047 |
| en_GU | CUOHL055 | 01200 |
| en_HK | CUOHL056 | 01047 |
| en_IE | CUOHL057 | 00924 |
| en_IE@euro | CUOHL057 | 00924 |

| <i>Table 88. z/OS Unicode Services locale standard source repository (continued)</i> | | |
|--|---------------|-----------------------------------|
| Locale name | Member | Default CCSID (if defined) |
| en_IE@preeuro | CUOHL058 | 00924 |
| en_IN | CUOHL059 | 01047 |
| en_JM | CUOHL05A | 01200 |
| en_MH | CUOHL05C | 01200 |
| en_MP | CUOHL05D | 01200 |
| en_MT | CUOHL05E | 01200 |
| en_MU | CUOHL05F | 01200 |
| en_NA | CUOHL060 | 01200 |
| en_NZ | CUOHL061 | 01047 |
| en_PH | CUOHL062 | 01047 |
| en_PK | CUOHL063 | 01200 |
| en_SG | CUOHL064 | 01047 |
| en_TT | CUOHL065 | 01200 |
| en_UM | CUOHL066 | 01200 |
| en_US | CUOHL067 | 01047 |
| en_US_POSIX | CUOHL068 | 01047 |
| en_US@euro | CUOHL069 | 01047 |
| en_VI | CUOHL06A | 01200 |
| en_ZA | CUOHL06B | 01047 |
| en_ZW | CUOHL06C | 01200 |
| es_AR | CUOHL06D | 01047 |
| es_BO | CUOHL06E | 01047 |
| es_CL | CUOHL06F | 01047 |
| es_CO | CUOHL070 | 01047 |
| es_CR | CUOHL071 | 01047 |
| es_DO | CUOHL072 | 01047 |
| es_EC | CUOHL073 | 01047 |
| es_ES | CUOHL074 | 01047 |
| es_ES@euro | CUOHL074 | 01047 |
| es_ES@preeuro | CUOHL075 | 01047 |
| es_GQ | CUOHL076 | 01200 |
| es_GT | CUOHL077 | 01047 |
| es_HN | CUOHL078 | 01047 |
| es_MX | CUOHL079 | 01047 |

Table 88. z/OS Unicode Services locale standard source repository (continued)

| Locale name | Member | Default CCSID (if defined) |
|---------------|----------|----------------------------|
| es_NI | CUOHL07A | 01047 |
| es_PA | CUOHL07B | 01047 |
| es_PE | CUOHL07C | 01047 |
| es_PR | CUOHL07D | 01047 |
| es_PY | CUOHL07E | 01047 |
| es_SV | CUOHL07F | 01047 |
| es_US | CUOHL080 | 01047 |
| es_UY | CUOHL081 | 01047 |
| es_VE | CUOHL082 | 01047 |
| es_VEO | CUOHL082 | 01047 |
| et_EE | CUOHL083 | 01122 |
| et_EE@euro | CUOHL084 | 01122 |
| et_EE@preeuro | CUOHL083 | 01122 |
| eu_ES | CUOHL085 | 01208 |
| fa_AF | CUOHL086 | 01200 |
| fa_IR | CUOHL087 | 01200 |
| ff_SN | CUOHL088 | 01200 |
| fi_FI | CUOHL089 | 01047 |
| fi_FI@euro | CUOHL089 | 01047 |
| fi_FI@preeuro | CUOHL08A | 01047 |
| fil_PH | CUOHL08B | 01200 |
| fo_FO | CUOHL08C | 01200 |
| fr_BE | CUOHL08D | 01047 |
| fr_BE@euro | CUOHL08D | 01047 |
| fr_BE@preeuro | CUOHL08E | 01047 |
| fr_BF | CUOHL08F | 01200 |
| fr_BI | CUOHL090 | 01200 |
| fr_BJ | CUOHL091 | 01200 |
| fr_BL | CUOHL092 | 01200 |
| fr_CA | CUOHL093 | 01047 |
| fr_CA@euro | CUOHL094 | 01047 |
| fr_CD | CUOHL095 | 01200 |
| fr_CF | CUOHL096 | 01200 |
| fr_CG | CUOHL097 | 01200 |

| <i>Table 88. z/OS Unicode Services locale standard source repository (continued)</i> | | |
|--|---------------|-----------------------------------|
| Locale name | Member | Default CCSID (if defined) |
| fr_CH | CUOHL098 | 01047 |
| fr_CH@euro | CUOHL099 | 01047 |
| fr_CI | CUOHL09A | 01200 |
| fr_CM | CUOHL09B | 01200 |
| fr_DJ | CUOHL09C | 01200 |
| fr_FR | CUOHL09D | 01047 |
| fr_FR@euro | CUOHL09D | 01047 |
| fr_FR@preeuro | CUOHL09E | 01047 |
| fr_GA | CUOHL09F | 01200 |
| fr_GN | CUOHL0A0 | 01200 |
| fr_GP | CUOHL0A1 | 01200 |
| fr_GQ | CUOHL0A2 | 01200 |
| fr_KM | CUOHL0A3 | 01200 |
| fr_LU | CUOHL0A4 | 00924 |
| fr_LU@euro | CUOHL0A4 | 00924 |
| fr_LU@preeuro | CUOHL0A5 | 00924 |
| fr_MC | CUOHL0A6 | 01200 |
| fr_MF | CUOHL0A7 | 01200 |
| fr_MG | CUOHL0A8 | 01200 |
| fr_ML | CUOHL0A9 | 01200 |
| fr_MQ | CUOHL0AA | 01200 |
| fr_NE | CUOHL0AB | 01200 |
| fr_RE | CUOHL0AC | 01200 |
| fr_RW | CUOHL0AD | 01200 |
| fr_SN | CUOHL0AE | 01200 |
| fr_TD | CUOHL0AF | 01200 |
| fr_TG | CUOHL0B0 | 01200 |
| fur_IT | CUOHL0B1 | 01200 |
| ga_IE | CUOHL0B2 | 01200 |
| gaa_GH | CUOHL0B3 | 01200 |
| gez_ER | CUOHL0B4 | 01200 |
| gez_ET | CUOHL0B5 | 01200 |
| gl_ES | CUOHL0B6 | 01208 |
| gsw_CH | CUOHL0B7 | 01200 |

Table 88. z/OS Unicode Services locale standard source repository (continued)

| Locale name | Member | Default CCSID (if defined) |
|---------------|----------|----------------------------|
| gu_IN | CUOHL0B8 | 01208 |
| guz_KE | CUOHL0B9 | 01200 |
| gv_GB | CUOHL0BA | 01200 |
| ha_Arab_NG | CUOHL0BB | 01200 |
| ha_Arab_SD | CUOHL0BC | 01200 |
| ha_Latn_GH | CUOHL0BD | 01200 |
| ha_Latn_NE | CUOHL0BE | 01200 |
| ha_Latn_NG | CUOHL0BF | 01200 |
| haw_US | CUOHL0C0 | 01200 |
| he_IL | CUOHL0C1 | 00424 |
| hi_IN | CUOHL0C2 | 01208 |
| hr_HR | CUOHL0C3 | 00870 |
| hu_HU | CUOHL0C4 | 00870 |
| hu_HU@euro | CUOHL0C5 | 00870 |
| hu_HU@preeuro | CUOHL0C4 | 00870 |
| hy_AM | CUOHL0C6 | 01208 |
| id_ID | CUOHL0C7 | 01047 |
| ig_NG | CUOHL0C8 | 01200 |
| ii_CN | CUOHL0C9 | 01200 |
| is_IS | CUOHL0CA | 01047 |
| is_IS@euro | CUOHL0CB | 01047 |
| it_CH | CUOHL0CC | 01047 |
| it_IT | CUOHL0CD | 01047 |
| it_IT@euro | CUOHL0CD | 01047 |
| it_IT@preeuro | CUOHL0CE | 01047 |
| iw_IL | CUOHL0C1 | 00424 |
| ja_JP | CUOHL0CF | 00939 |
| jmc_TZ | CUOHL0D0 | 01200 |
| ka_GE | CUOHL0D1 | 01208 |
| kab_DZ | CUOHL0D2 | 01200 |
| kaj_NG | CUOHL0D3 | 01200 |
| kam_KE | CUOHL0D4 | 01200 |
| kcg_NG | CUOHL0D5 | 01200 |
| kde_TZ | CUOHL0D6 | 01200 |

| <i>Table 88. z/OS Unicode Services locale standard source repository (continued)</i> | | |
|--|---------------|-----------------------------------|
| Locale name | Member | Default CCSID (if defined) |
| kea_CV | CUOHL0D7 | 01200 |
| kfo_CI | CUOHL0D8 | 01200 |
| khq_ML | CUOHL0D9 | 01200 |
| ki_KE | CUOHL0DA | 01200 |
| kk_Cyrl_KZ | CUOHL0DB | 01200 |
| kk_KZ | CUOHL0DC | 01208 |
| kl_GL | CUOHL0DD | 01200 |
| kl_n_KE | CUOHL0DE | 01200 |
| km_KH | CUOHL0DF | 01200 |
| kn_IN | CUOHL0E0 | 01208 |
| ko_KR | CUOHL0E1 | 00933 |
| kok_IN | CUOHL0E2 | 01200 |
| kpe_GN | CUOHL0E3 | 01200 |
| kpe_LR | CUOHL0E4 | 01200 |
| ksb_TZ | CUOHL0E5 | 01200 |
| ksh_DE | CUOHL0E6 | 01200 |
| ku_Arab_IQ | CUOHL0E7 | 01200 |
| ku_Arab_IR | CUOHL0E8 | 01200 |
| ku_Latn_SY | CUOHL0E9 | 01200 |
| ku_Latn_TR | CUOHL0EA | 01200 |
| kw_GB | CUOHL0EB | 01200 |
| ky_KG | CUOHL0EC | 01200 |
| lag_TZ | CUOHL0ED | 01200 |
| lg_UG | CUOHL0EE | 01200 |
| ln_CD | CUOHL0EF | 01200 |
| ln_CG | CUOHL0F0 | 01200 |
| lo_LA | CUOHL0F1 | 01200 |
| lt_LT | CUOHL0F2 | 01112 |
| lt_LT@euro | CUOHL0F3 | 01112 |
| lt_LT@preeuro | CUOHL0F2 | 01112 |
| luo_KE | CUOHL0F4 | 01200 |
| luy_KE | CUOHL0F5 | 01200 |
| lv_LV | CUOHL0F6 | 01112 |
| lv_LV@euro | CUOHL0F7 | 01112 |

Table 88. z/OS Unicode Services locale standard source repository (continued)

| Locale name | Member | Default CCSID (if defined) |
|---------------|----------|----------------------------|
| lv_LV@preeuro | CUOHL0F6 | 01112 |
| mas_KE | CUOHL0F8 | 01200 |
| mas_TZ | CUOHL0F9 | 01200 |
| mer_KE | CUOHL0FA | 01200 |
| mfe_MU | CUOHL0FB | 01200 |
| mg_MG | CUOHL0FC | 01200 |
| mi_NZ | CUOHL0FD | 01200 |
| mk_MK | CUOHL0FE | 01025 |
| ml_IN | CUOHL0FF | 01200 |
| mn_Cyrl_MN | CUOHL100 | 01200 |
| mn_Mong_CN | CUOHL101 | 01200 |
| mr_IN | CUOHL102 | 01208 |
| ms_BN | CUOHL103 | 01200 |
| ms_MY | CUOHL104 | 01047 |
| mt_MT | CUOHL105 | 01208 |
| mt_MT@euro | CUOHL105 | 01208 |
| mt_MT@preeuro | CUOHL106 | 01208 |
| my_MM | CUOHL107 | 01200 |
| naq_NA | CUOHL108 | 01200 |
| nb_NO | CUOHL109 | 01208 |
| nd_ZW | CUOHL10A | 01200 |
| nds_DE | CUOHL10B | 01200 |
| ne_IN | CUOHL10C | 01200 |
| ne_NP | CUOHL10D | 01200 |
| nl_BE | CUOHL10E | 01047 |
| nl_BE@euro | CUOHL10E | 01047 |
| nl_BE@preeuro | CUOHL10F | 01047 |
| nl_NL | CUOHL110 | 01047 |
| nl_NL@euro | CUOHL110 | 01047 |
| nl_NL@preeuro | CUOHL111 | 01047 |
| nn_NO | CUOHL112 | 01208 |
| no_NO@euro | CUOHL113 | 01047 |
| nr_ZA | CUOHL114 | 01200 |
| nso_ZA | CUOHL115 | 01200 |

| <i>Table 88. z/OS Unicode Services locale standard source repository (continued)</i> | | |
|--|---------------|-----------------------------------|
| Locale name | Member | Default CCSID (if defined) |
| ny_MW | CUOHL116 | 01200 |
| nyn_UG | CUOHL117 | 01200 |
| oc_FR | CUOHL118 | 01200 |
| om_ET | CUOHL119 | 01200 |
| om_KE | CUOHL11A | 01200 |
| or_IN | CUOHL11B | 01200 |
| pa_Arab_PK | CUOHL11C | 01200 |
| pa_Guru_IN | CUOHL11D | 01208 |
| pa_IN | CUOHL11D | 01208 |
| pl_PL | CUOHL11E | 00870 |
| pl_PL@euro | CUOHL11F | 00870 |
| pl_PL@preeuro | CUOHL11E | 00870 |
| posix | CUOHL068 | 01047 |
| ps_AF | CUOHL120 | 01200 |
| pt_AO | CUOHL121 | 01200 |
| pt_BR | CUOHL122 | 01047 |
| pt_BR@euro | CUOHL123 | 01047 |
| pt_GW | CUOHL124 | 01200 |
| pt_MZ | CUOHL125 | 01200 |
| pt_PT | CUOHL126 | 01047 |
| pt_PT@euro | CUOHL126 | 01047 |
| pt_PT@preeuro | CUOHL127 | 01047 |
| rm_CH | CUOHL128 | 01200 |
| ro_MD | CUOHL129 | 01200 |
| ro_RO | CUOHL12A | 00870 |
| ro_RO@euro | CUOHL12B | 00870 |
| ro_RO@preeuro | CUOHL12A | 00870 |
| rof_TZ | CUOHL12C | 01200 |
| ru_MD | CUOHL12D | 01200 |
| ru_RU | CUOHL12E | 01025 |
| ru_UA | CUOHL12F | 01200 |
| rw_RW | CUOHL130 | 01200 |
| rwk_TZ | CUOHL131 | 01200 |
| sa_IN | CUOHL132 | 01200 |

Table 88. z/OS Unicode Services locale standard source repository (continued)

| Locale name | Member | Default CCSID (if defined) |
|---------------|----------|----------------------------|
| saq_KE | CUOHL133 | 01200 |
| se_FI | CUOHL134 | 01200 |
| se_NO | CUOHL135 | 01200 |
| seh_MZ | CUOHL136 | 01200 |
| ses_ML | CUOHL137 | 01200 |
| sg_CF | CUOHL138 | 01200 |
| shi_Latn_MA | CUOHL13A | 01200 |
| shi_Tfng_MA | CUOHL13B | 01200 |
| si_LK | CUOHL13C | 01200 |
| sid_ET | CUOHL13D | 01200 |
| sk_SK | CUOHL13E | 00870 |
| sk_SK@euro | CUOHL13E | 00870 |
| sk_SK@preeuro | CUOHL13F | 00870 |
| sl_SI | CUOHL140 | 00870 |
| sl_SI@euro | CUOHL140 | 00870 |
| sl_SI@preeuro | CUOHL141 | 00870 |
| sn_ZW | CUOHL142 | 01200 |
| so_DJ | CUOHL143 | 01200 |
| so_ET | CUOHL144 | 01200 |
| so_KE | CUOHL145 | 01200 |
| so_SO | CUOHL146 | 01200 |
| sq_AL | CUOHL147 | 01047 |
| sq_AL@euro | CUOHL148 | 01047 |
| sr_CS | CUOHL14B | 01208 |
| sr_Cyrl_BA | CUOHL149 | 01200 |
| sr_Cyrl_ME | CUOHL14A | 01200 |
| sr_Cyrl_RS | CUOHL14B | 01208 |
| sr_Latn_BA | CUOHL14C | 01200 |
| sr_Latn_ME | CUOHL14D | 01200 |
| sr_Latn_RS | CUOHL14E | 01208 |
| sr_RS | CUOHL14E | 01208 |
| ss_SZ | CUOHL150 | 01200 |
| ss_ZA | CUOHL151 | 01200 |
| ssy_ER | CUOHL152 | 01200 |

| <i>Table 88. z/OS Unicode Services locale standard source repository (continued)</i> | | |
|--|---------------|-----------------------------------|
| Locale name | Member | Default CCSID (if defined) |
| st_LS | CUOHL153 | 01200 |
| st_ZA | CUOHL154 | 01200 |
| sv_FI | CUOHL155 | 01200 |
| sv_SE | CUOHL156 | 01047 |
| sv_SE@euro | CUOHL157 | 01047 |
| sv_SE@preeuro | CUOHL156 | 01047 |
| sw_KE | CUOHL158 | 01208 |
| sw_TZ | CUOHL159 | 01208 |
| syr_SY | CUOHL15A | 01200 |
| ta_IN | CUOHL15B | 01208 |
| ta_LK | CUOHL15C | 01200 |
| te_IN | CUOHL15D | 01208 |
| teo_KE | CUOHL15E | 01200 |
| teo_UG | CUOHL15F | 01200 |
| tg_Cyrl_TJ | CUOHL160 | 01200 |
| th_TH | CUOHL161 | 00838 |
| ti_ER | CUOHL162 | 01200 |
| ti_ET | CUOHL163 | 01200 |
| tig_ER | CUOHL164 | 01200 |
| tn_ZA | CUOHL165 | 01200 |
| to_TO | CUOHL166 | 01200 |
| tr_TR | CUOHL167 | 01026 |
| trv_TW | CUOHL169 | 01200 |
| ts_ZA | CUOHL16A | 01200 |
| tt_RU | CUOHL16B | 01200 |
| tzm_Latn_MA | CUOHL16C | 01200 |
| ug_Arab_CN | CUOHL16D | 01200 |
| uk_UA | CUOHL16E | 01123 |
| ur_IN | CUOHL16F | 01200 |
| ur_PK | CUOHL170 | 01200 |
| uz_Arab_AF | CUOHL171 | 01200 |
| uz_Cyrl_UZ | CUOHL172 | 01200 |
| uz_Latn_UZ | CUOHL173 | 01200 |
| ve_ZA | CUOHL174 | 01200 |

Table 88. z/OS Unicode Services locale standard source repository (continued)

| Locale name | Member | Default CCSID (if defined) |
|-------------|----------|----------------------------|
| vi_VN | CUOHL175 | 01208 |
| vun_TZ | CUOHL176 | 01200 |
| wal_ET | CUOHL177 | 01200 |
| wo_Latn_SN | CUOHL178 | 01200 |
| xh_ZA | CUOHL179 | 01200 |
| xog_UG | CUOHL17A | 01200 |
| yo_NG | CUOHL17B | 01200 |
| zh_CN | CUOHL17C | 00935 |
| zh_Hans_CN | CUOHL17C | 00935 |
| zh_Hans_HK | CUOHL17D | 01200 |
| zh_Hans_MO | CUOHL17E | 01200 |
| zh_Hans_SG | CUOHL17F | 01208 |
| zh_Hant_HK | CUOHL180 | 01208 |
| zh_Hant_MO | CUOHL181 | 01200 |
| zh_Hant_TW | CUOHL182 | 00937 |
| zh_HK | CUOHL180 | 01208 |
| zh_SG | CUOHL17F | 01208 |
| zh_TW | CUOHL182 | 00937 |
| zu_ZA | CUOHL186 | 01208 |

Notes:

1. The locale name is used by the z/OS Unicode Services dynamic locale service API and console commands. The member of the locales in the SYS1.SCUNTLB data set is listed in the tables for reference information only.
2. The locale support being provided by z/OS Unicode Services is a superset of that currently provided by the C/C++ Run-time Library.

Adding and removing locales to the z/OS Unicode Services environment

z/OS Unicode Services maintains the storage for locale objects. Once created, the locale objects remain until the next IPL or they are deleted via the SETUNI DELETE,BLDLOCALE command. See [z/OS MVS System Commands](#) for additional information. Because the locale objects exist outside of C/C++ Run-time storage, they are available to any z/OS Unicode Services user.

Locale objects can be added to the z/OS Unicode Services environment in any of these ways:

- By calling the z/OS Unicode Services dynamic locale service.
- By adding new statements to the CUNUNlxx parmlib member that allows users to say what locales to load during the system IPL. See [z/OS MVS Initialization and Tuning Reference](#) for additional information.
- With the SETUNI ADD,BLDLOCALE command. See [z/OS MVS System Commands](#) for additional information.

Euro and pre-euro support

An @euro codeset modifier or an @preeuro codeset modifier on the locale name is used to support the euro and pre-euro versions. This is analogous to the current support provided by the C/C++ Run-time Library. For example, a pre-euro version of the bg_BG locale would be named bg_BG@preeuro.

Language Environment C/C++ Runtime Library compatible locale support

Compatible support for the following Language Environment C/C++ Runtime Library locale source is shipped in the SYS1.SCUNTL data set.

Note: The Language Environment-compatible locale source is named the same as in the C/C++ Runtime Library, with the exception of a .le extension to the name. So, for example, if you want US English locale data that was compatible with the En_US.IBM-1047 C/C++ Runtime Library locale, call the z/OS Unicode Services dynamic locale service using En_US.IBM-1047.le as the locale name.

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| af_ZA.UTF-8.le | CUOHL210 | 01208 |
| Af_ZA.UTF-8.le | CUOHL210 | 01208 |
| Ar_AA.IBM-425.le | CUOHL187 | |
| Ar_AA.le | CUOHL187 | 00425 |
| az_AZ.UTF-8.le | CUOHL211 | 01208 |
| Az_AZ.UTF-8.le | CUOHL211 | 01208 |
| Be_BY.IBM-1025.le | CUOHL188 | |
| Be_BY.IBM-1154.le | CUOHL189 | |
| be_BY.ISO8859-5.le | CUOHL212 | |
| Be_BY.ISO8859-5.le | CUOHL212 | |
| Bg_BG.IBM-1025.le | CUOHL18A | |
| Bg_BG.IBM-1154.le | CUOHL18B | |
| Bg_BG.IBM-1154@euro.le | CUOHL18C | |
| Bg_BG.IBM-1154@preeuro.le | CUOHL18B | |
| Bg_BG.le | CUOHL18A | 01025 |
| bg_BG.UTF-8.le | CUOHL213 | |
| Bg_BG.UTF-8.le | CUOHL213 | |
| bg_BG.UTF-8@euro.le | CUOHL214 | |
| Bg_BG.UTF-8@euro.le | CUOHL214 | |
| bg_BG.UTF-8@preeuro.le | CUOHL213 | |
| Bg_BG.UTF-8@preeuro.le | CUOHL213 | |
| bn_IN.UTF-8.le | CUOHL215 | 01208 |
| Bn_IN.UTF-8.le | CUOHL215 | 01208 |
| C.le | CUOHL278 | 01047 |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| Ca_ES.IBM-924.le | CUOHL18D | |
| Ca_ES.IBM-924@euro.le | CUOHL18D | |
| Ca_ES.IBM-924@preeuro.le | CUOHL18E | |
| ca_ES.UTF-8.le | CUOHL216 | |
| Ca_ES.UTF-8.le | CUOHL216 | |
| ca_ES.UTF-8@euro.le | CUOHL216 | |
| Ca_ES.UTF-8@euro.le | CUOHL216 | |
| ca_ES.UTF-8@preeuro.le | CUOHL217 | |
| Ca_ES.UTF-8@preeuro.le | CUOHL217 | |
| Cs_CZ.IBM-1153.le | CUOHL191 | |
| Cs_CZ.IBM-1153@euro.le | CUOHL190 | |
| Cs_CZ.IBM-1153@preeuro.le | CUOHL191 | |
| Cs_CZ.IBM-1165.le | CUOHL193 | |
| Cs_CZ.IBM-1165@euro.le | CUOHL192 | |
| Cs_CZ.IBM-1165@preeuro.le | CUOHL193 | |
| Cs_CZ.IBM-870.le | CUOHL18F | |
| cs_CZ.ISO8859-2.le | CUOHL218 | |
| Cs_CZ.ISO8859-2.le | CUOHL218 | |
| Cs_CZ.le | CUOHL18F | 00870 |
| cs_CZ.UTF-8.le | CUOHL219 | |
| Cs_CZ.UTF-8.le | CUOHL219 | |
| cs_CZ.UTF-8@euro.le | CUOHL2B1 | |
| Cs_CZ.UTF-8@euro.le | CUOHL2B1 | |
| cs_CZ.UTF-8@preeuro.le | CUOHL219 | |
| Cs_CZ.UTF-8@preeuro.le | CUOHL219 | |
| cy_GB.UTF-8.le | CUOHL2B2 | 01208 |
| Cy_GB.UTF-8.le | CUOHL2B2 | 01208 |
| cy_GB.UTF-8@euro.le | CUOHL2B3 | 01208 |
| Cy_GB.UTF-8@euro.le | CUOHL2B3 | 01208 |
| Da_DK.IBM-1047.le | CUOHL194 | |
| Da_DK.IBM-1142.le | CUOHL195 | |
| Da_DK.IBM-1142@euro.le | CUOHL196 | |
| Da_DK.IBM-277.le | CUOHL194 | |
| Da_DK.IBM-924.le | CUOHL197 | |
| Da_DK.IBM-924@euro.le | CUOHL198 | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|---------------|-----------------------------------|
| da_DK.ISO8859-1.le | CUOHL2B4 | |
| Da_DK.ISO8859-1.le | CUOHL2B4 | |
| Da_DK.le | CUOHL194 | 01047 |
| da_DK.UTF-8.le | CUOHL2B5 | |
| Da_DK.UTF-8.le | CUOHL2B5 | |
| da_DK.UTF-8@euro.le | CUOHL2B6 | |
| Da_DK.UTF-8@euro.le | CUOHL2B6 | |
| De_AT.IBM-924.le | CUOHL199 | |
| De_AT.IBM-924@euro.le | CUOHL199 | |
| De_AT.IBM-924@preeuro.le | CUOHL19A | |
| de_AT.UTF-8.le | CUOHL2B7 | |
| De_AT.UTF-8.le | CUOHL2B7 | |
| de_AT.UTF-8@euro.le | CUOHL2B7 | |
| De_AT.UTF-8@euro.le | CUOHL2B7 | |
| de_AT.UTF-8@preeuro.le | CUOHL2B8 | |
| De_AT.UTF-8@preeuro.le | CUOHL2B8 | |
| De_CH.IBM-1047.le | CUOHL19B | |
| De_CH.IBM-1148.le | CUOHL19C | |
| De_CH.IBM-1148@euro.le | CUOHL19D | |
| De_CH.IBM-500.le | CUOHL19B | |
| de_CH.ISO8859-1.le | CUOHL2B9 | |
| De_CH.ISO8859-1.le | CUOHL2B9 | |
| De_CH.le | CUOHL19B | 01047 |
| de_CH.UTF-8.le | CUOHL2BA | |
| De_CH.UTF-8.le | CUOHL2BA | |
| De_DE.IBM-1047.le | CUOHL19E | |
| De_DE.IBM-1141.le | CUOHL19F | |
| De_DE.IBM-1141@euro.le | CUOHL19F | |
| De_DE.IBM-1141@preeuro.le | CUOHL1A0 | |
| De_DE.IBM-273.le | CUOHL19E | |
| De_DE.IBM-924.le | CUOHL1A1 | |
| De_DE.IBM-924@euro.le | CUOHL1A1 | |
| De_DE.IBM-924@preeuro.le | CUOHL1A2 | |
| de_DE.ISO8859-1.le | CUOHL2BB | |
| De_DE.ISO8859-1.le | CUOHL2BB | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| De_DE.le | CUOHL19E | 01047 |
| de_DE.UTF-8.le | CUOHL2BC | |
| De_DE.UTF-8.le | CUOHL2BC | |
| de_DE.UTF-8@euro.le | CUOHL2BC | |
| De_DE.UTF-8@euro.le | CUOHL2BC | |
| de_DE.UTF-8@preeuro.le | CUOHL2BD | |
| De_DE.UTF-8@preeuro.le | CUOHL2BD | |
| De_LU.IBM-924.le | CUOHL1A3 | |
| De_LU.IBM-924@euro.le | CUOHL1A3 | |
| De_LU.IBM-924@preeuro.le | CUOHL1A4 | |
| de_LU.UTF-8.le | CUOHL2BE | |
| De_LU.UTF-8.le | CUOHL2BE | |
| de_LU.UTF-8@euro.le | CUOHL2BE | |
| De_LU.UTF-8@euro.le | CUOHL2BE | |
| de_LU.UTF-8@preeuro.le | CUOHL2BF | |
| De_LU.UTF-8@preeuro.le | CUOHL2BF | |
| El_GR.IBM-4971.le | CUOHL1A6 | |
| El_GR.IBM-4971@euro.le | CUOHL1A6 | |
| El_GR.IBM-4971@preeuro.le | CUOHL1A7 | |
| El_GR.IBM-875.le | CUOHL1A5 | |
| el_GR.ISO8859-7.le | CUOHL2C0 | |
| El_GR.ISO8859-7.le | CUOHL2C0 | |
| El_GR.le | CUOHL1A5 | 00875 |
| el_GR.UTF-8.le | CUOHL2C1 | |
| El_GR.UTF-8.le | CUOHL2C1 | |
| el_GR.UTF-8@euro.le | CUOHL2C1 | |
| El_GR.UTF-8@euro.le | CUOHL2C1 | |
| el_GR.UTF-8@preeuro.le | CUOHL2C2 | |
| El_GR.UTF-8@preeuro.le | CUOHL2C2 | |
| En_AU.IBM-1047.le | CUOHL1A8 | |
| en_AU.ISO8859-1.le | CUOHL2C3 | |
| En_AU.ISO8859-1.le | CUOHL2C3 | |
| En_BE.IBM-924.le | CUOHL1A9 | |
| En_BE.IBM-924@euro.le | CUOHL1A9 | |
| En_BE.IBM-924@preeuro.le | CUOHL1AA | |

| Locale name | Member | Default CCSID (if defined) |
|--------------------------|---------------|-----------------------------------|
| en_BE.UTF-8.le | CUOHL2C4 | |
| En_BE.UTF-8.le | CUOHL2C4 | |
| en_BE.UTF-8@euro.le | CUOHL2C4 | |
| En_BE.UTF-8@euro.le | CUOHL2C4 | |
| en_BE.UTF-8@preeuro.le | CUOHL2C5 | |
| En_BE.UTF-8@preeuro.le | CUOHL2C5 | |
| En_CA.IBM-037.le | CUOHL1AD | |
| En_CA.IBM-1047.le | CUOHL1AC | |
| En_CA.IBM-1140.le | CUOHL1AB | |
| en_CA.IBM-923.le | CUOHL2C7 | |
| En_CA.IBM-923.le | CUOHL2C7 | |
| En_CA.IBM-924.le | CUOHL1AE | |
| en_CA.ISO8859-1.le | CUOHL2C6 | |
| En_CA.ISO8859-1.le | CUOHL2C6 | |
| En_GB.IBM-1047.le | CUOHL1AF | |
| En_GB.IBM-1146.le | CUOHL1B0 | |
| En_GB.IBM-1146@euro.le | CUOHL1B1 | |
| En_GB.IBM-285.le | CUOHL1AF | |
| En_GB.IBM-924.le | CUOHL1B2 | |
| En_GB.IBM-924@euro.le | CUOHL1B3 | |
| en_GB.ISO8859-1.le | CUOHL2C8 | |
| En_GB.ISO8859-1.le | CUOHL2C8 | |
| En_GB.le | CUOHL1AF | 01047 |
| en_GB.UTF-8.le | CUOHL2C9 | |
| En_GB.UTF-8.le | CUOHL2C9 | |
| en_GB.UTF-8@euro.le | CUOHL2CA | |
| En_GB.UTF-8@euro.le | CUOHL2CA | |
| En_HK.IBM-1047.le | CUOHL1B4 | |
| en_HK.ISO8859-1.le | CUOHL2CB | |
| En_HK.ISO8859-1.le | CUOHL2CB | |
| En_IE.IBM-924.le | CUOHL1B5 | |
| En_IE.IBM-924@euro.le | CUOHL1B5 | |
| En_IE.IBM-924@preeuro.le | CUOHL1B6 | |
| en_IE.UTF-8.le | CUOHL2CC | |
| En_IE.UTF-8.le | CUOHL2CC | |

| Locale name | Member | Default CCSID (if defined) |
|------------------------|----------|----------------------------|
| en_IE.UTF-8@euro.le | CUOHL2CC | |
| En_IE.UTF-8@euro.le | CUOHL2CC | |
| en_IE.UTF-8@preeuro.le | CUOHL2CD | |
| En_IE.UTF-8@preeuro.le | CUOHL2CD | |
| En_IN.IBM-1047.le | CUOHL1B7 | |
| en_IN.ISO8859-1.le | CUOHL2CE | |
| En_IN.ISO8859-1.le | CUOHL2CE | |
| En_JP.IBM-1027.le | CUOHL1B8 | |
| En_JP.IBM-5123.le | CUOHL1B9 | |
| En_JP.le | CUOHL1B8 | 01027 |
| En_NZ.IBM-1047.le | CUOHL1BA | |
| en_NZ.ISO8859-1.le | CUOHL2CF | |
| En_NZ.ISO8859-1.le | CUOHL2CF | |
| En_PH.IBM-1047.le | CUOHL1BB | |
| en_PH.ISO8859-1.le | CUOHL2D0 | |
| En_PH.ISO8859-1.le | CUOHL2D0 | |
| En_SG.IBM-1047.le | CUOHL1BC | |
| en_SG.ISO8859-1.le | CUOHL2D1 | |
| En_SG.ISO8859-1.le | CUOHL2D1 | |
| En_US.IBM-037.le | CUOHL1BD | |
| En_US.IBM-1047.le | CUOHL1BD | |
| En_US.IBM-1140.le | CUOHL1BE | |
| En_US.IBM-1140@euro.le | CUOHL1BF | |
| en_US.ISO8859-1.le | CUOHL2D2 | |
| En_US.ISO8859-1.le | CUOHL2D2 | |
| En_US.le | CUOHL1BD | 01047 |
| en_US.UTF-8.le | CUOHL2D3 | |
| En_US.UTF-8.le | CUOHL2D3 | |
| En_ZA.IBM-037.le | CUOHL1C0 | |
| En_ZA.IBM-1047.le | CUOHL1C1 | |
| En_ZA.IBM-1140.le | CUOHL1C2 | |
| en_ZA.IBM-923.le | CUOHL2D5 | |
| En_ZA.IBM-923.le | CUOHL2D5 | |
| En_ZA.IBM-924.le | CUOHL1C3 | |
| en_ZA.ISO8859-1.le | CUOHL2D4 | |

| Locale name | Member | Default CCSID (if defined) |
|--------------------|---------------|-----------------------------------|
| En_ZA.ISO8859-1.le | CUOHL2D4 | |
| Es_AR.IBM-1047.le | CUOHL1C4 | |
| Es_AR.IBM-1145.le | CUOHL1C5 | |
| Es_AR.IBM-284.le | CUOHL1C6 | |
| es_AR.IBM-923.le | CUOHL2D7 | |
| Es_AR.IBM-923.le | CUOHL2D7 | |
| Es_AR.IBM-924.le | CUOHL1C7 | |
| es_AR.ISO8859-1.le | CUOHL2D6 | |
| Es_AR.ISO8859-1.le | CUOHL2D6 | |
| Es_BO.IBM-1047.le | CUOHL1C8 | |
| Es_BO.IBM-1145.le | CUOHL1C9 | |
| Es_BO.IBM-284.le | CUOHL1CA | |
| es_BO.IBM-923.le | CUOHL2D9 | |
| Es_BO.IBM-923.le | CUOHL2D9 | |
| Es_BO.IBM-924.le | CUOHL1CB | |
| es_BO.ISO8859-1.le | CUOHL2D8 | |
| Es_BO.ISO8859-1.le | CUOHL2D8 | |
| Es_CL.IBM-1047.le | CUOHL1CC | |
| Es_CL.IBM-1145.le | CUOHL1CD | |
| Es_CL.IBM-284.le | CUOHL1CE | |
| es_CL.IBM-923.le | CUOHL2DB | |
| Es_CL.IBM-923.le | CUOHL2DB | |
| Es_CL.IBM-924.le | CUOHL1CF | |
| es_CL.ISO8859-1.le | CUOHL2DA | |
| Es_CL.ISO8859-1.le | CUOHL2DA | |
| Es_CO.IBM-1047.le | CUOHL1D0 | |
| Es_CO.IBM-1145.le | CUOHL1D1 | |
| Es_CO.IBM-284.le | CUOHL1D2 | |
| es_CO.IBM-923.le | CUOHL2DD | |
| Es_CO.IBM-923.le | CUOHL2DD | |
| Es_CO.IBM-924.le | CUOHL1D3 | |
| es_CO.ISO8859-1.le | CUOHL2DC | |
| Es_CO.ISO8859-1.le | CUOHL2DC | |
| Es_CR.IBM-1047.le | CUOHL1D4 | |
| Es_CR.IBM-1145.le | CUOHL1D5 | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| Es_CR.IBM-284.le | CUOHL1D6 | |
| es_CR.IBM-923.le | CUOHL2DF | |
| Es_CR.IBM-923.le | CUOHL2DF | |
| Es_CR.IBM-924.le | CUOHL1D7 | |
| es_CR.ISO8859-1.le | CUOHL2DE | |
| Es_CR.ISO8859-1.le | CUOHL2DE | |
| Es_DO.IBM-1047.le | CUOHL1D8 | |
| Es_DO.IBM-1145.le | CUOHL1D9 | |
| Es_DO.IBM-284.le | CUOHL1DA | |
| es_DO.IBM-923.le | CUOHL2E1 | |
| Es_DO.IBM-923.le | CUOHL2E1 | |
| Es_DO.IBM-924.le | CUOHL1DB | |
| es_DO.ISO8859-1.le | CUOHL2E0 | |
| Es_DO.ISO8859-1.le | CUOHL2E0 | |
| Es_EC.IBM-1047.le | CUOHL1DC | |
| Es_EC.IBM-1145.le | CUOHL1DD | |
| Es_EC.IBM-284.le | CUOHL1DE | |
| es_EC.IBM-923.le | CUOHL2E3 | |
| Es_EC.IBM-923.le | CUOHL2E3 | |
| Es_EC.IBM-924.le | CUOHL1DF | |
| es_EC.ISO8859-1.le | CUOHL2E2 | |
| Es_EC.ISO8859-1.le | CUOHL2E2 | |
| Es_ES.IBM-1047.le | CUOHL1E4 | |
| Es_ES.IBM-1145.le | CUOHL1E0 | |
| Es_ES.IBM-1145@euro.le | CUOHL1E0 | |
| Es_ES.IBM-1145@preeuro.le | CUOHL1E1 | |
| Es_ES.IBM-284.le | CUOHL1E4 | |
| Es_ES.IBM-924.le | CUOHL1E2 | |
| Es_ES.IBM-924@euro.le | CUOHL1E2 | |
| Es_ES.IBM-924@preeuro.le | CUOHL1E3 | |
| es_ES.ISO8859-1.le | CUOHL2E4 | |
| Es_ES.ISO8859-1.le | CUOHL2E4 | |
| Es_ES.le | CUOHL1E4 | 01047 |
| es_ES.UTF-8.le | CUOHL2E5 | |
| Es_ES.UTF-8.le | CUOHL2E5 | |

| Locale name | Member | Default CCSID (if defined) |
|------------------------|---------------|-----------------------------------|
| es_ES.UTF-8@euro.le | CUOHL2E5 | |
| Es_ES.UTF-8@euro.le | CUOHL2E5 | |
| es_ES.UTF-8@preeuro.le | CUOHL2E6 | |
| Es_ES.UTF-8@preeuro.le | CUOHL2E6 | |
| Es_GT.IBM-1047.le | CUOHL1E5 | |
| Es_GT.IBM-1145.le | CUOHL1E6 | |
| Es_GT.IBM-284.le | CUOHL1E7 | |
| es_GT.IBM-923.le | CUOHL2E8 | |
| Es_GT.IBM-923.le | CUOHL2E8 | |
| Es_GT.IBM-924.le | CUOHL1E8 | |
| es_GT.ISO8859-1.le | CUOHL2E7 | |
| Es_GT.ISO8859-1.le | CUOHL2E7 | |
| Es_HN.IBM-1047.le | CUOHL1E9 | |
| Es_HN.IBM-1145.le | CUOHL1EA | |
| Es_HN.IBM-284.le | CUOHL1EB | |
| es_HN.IBM-923.le | CUOHL2EA | |
| Es_HN.IBM-923.le | CUOHL2EA | |
| Es_HN.IBM-924.le | CUOHL1EC | |
| es_HN.ISO8859-1.le | CUOHL2E9 | |
| Es_HN.ISO8859-1.le | CUOHL2E9 | |
| Es_MX.IBM-1047.le | CUOHL1ED | |
| Es_MX.IBM-1145.le | CUOHL1EE | |
| Es_MX.IBM-284.le | CUOHL1EF | |
| es_MX.IBM-923.le | CUOHL2EC | |
| Es_MX.IBM-923.le | CUOHL2EC | |
| Es_MX.IBM-924.le | CUOHL1F0 | |
| es_MX.ISO8859-1.le | CUOHL2EB | |
| Es_MX.ISO8859-1.le | CUOHL2EB | |
| Es_NI.IBM-1047.le | CUOHL1F1 | |
| Es_NI.IBM-1145.le | CUOHL1F2 | |
| Es_NI.IBM-284.le | CUOHL1F3 | |
| es_NI.IBM-923.le | CUOHL2EE | |
| Es_NI.IBM-923.le | CUOHL2EE | |
| Es_NI.IBM-924.le | CUOHL1F4 | |
| es_NI.ISO8859-1.le | CUOHL2ED | |

| Locale name | Member | Default CCSID (if defined) |
|--------------------|----------|----------------------------|
| Es_NI.ISO8859-1.le | CUOHL2ED | |
| Es_PA.IBM-1047.le | CUOHL1F5 | |
| Es_PA.IBM-1145.le | CUOHL1F6 | |
| Es_PA.IBM-284.le | CUOHL1F7 | |
| es_PA.IBM-923.le | CUOHL2F0 | |
| Es_PA.IBM-923.le | CUOHL2F0 | |
| Es_PA.IBM-924.le | CUOHL1F8 | |
| es_PA.ISO8859-1.le | CUOHL2EF | |
| Es_PA.ISO8859-1.le | CUOHL2EF | |
| Es_PE.IBM-1047.le | CUOHL1F9 | |
| Es_PE.IBM-1145.le | CUOHL1FA | |
| Es_PE.IBM-284.le | CUOHL1FB | |
| es_PE.IBM-923.le | CUOHL2F2 | |
| Es_PE.IBM-923.le | CUOHL2F2 | |
| Es_PE.IBM-924.le | CUOHL1FC | |
| es_PE.ISO8859-1.le | CUOHL2F1 | |
| Es_PE.ISO8859-1.le | CUOHL2F1 | |
| Es_PR.IBM-1047.le | CUOHL1FD | |
| Es_PR.IBM-1145.le | CUOHL1FE | |
| Es_PR.IBM-284.le | CUOHL1FF | |
| es_PR.IBM-923.le | CUOHL2F4 | |
| Es_PR.IBM-923.le | CUOHL2F4 | |
| Es_PR.IBM-924.le | CUOHL200 | |
| es_PR.ISO8859-1.le | CUOHL2F3 | |
| Es_PR.ISO8859-1.le | CUOHL2F3 | |
| Es_PY.IBM-1047.le | CUOHL201 | |
| Es_PY.IBM-1145.le | CUOHL202 | |
| Es_PY.IBM-284.le | CUOHL203 | |
| es_PY.IBM-923.le | CUOHL2F6 | |
| Es_PY.IBM-923.le | CUOHL2F6 | |
| Es_PY.IBM-924.le | CUOHL204 | |
| es_PY.ISO8859-1.le | CUOHL2F5 | |
| Es_PY.ISO8859-1.le | CUOHL2F5 | |
| Es_SV.IBM-1047.le | CUOHL205 | |
| Es_SV.IBM-1145.le | CUOHL206 | |

| Locale name | Member | Default CCSID (if defined) |
|--------------------|---------------|-----------------------------------|
| Es_SV.IBM-284.le | CUOHL207 | |
| es_SV.IBM-923.le | CUOHL2F8 | |
| Es_SV.IBM-923.le | CUOHL2F8 | |
| Es_SV.IBM-924.le | CUOHL208 | |
| es_SV.ISO8859-1.le | CUOHL2F7 | |
| Es_SV.ISO8859-1.le | CUOHL2F7 | |
| Es_US.IBM-1047.le | CUOHL209 | |
| Es_US.IBM-1145.le | CUOHL20A | |
| Es_US.IBM-284.le | CUOHL20B | |
| es_US.IBM-923.le | CUOHL2FA | |
| Es_US.IBM-923.le | CUOHL2FA | |
| Es_US.IBM-924.le | CUOHL20C | |
| es_US.ISO8859-1.le | CUOHL2F9 | |
| Es_US.ISO8859-1.le | CUOHL2F9 | |
| Es_UY.IBM-1047.le | CUOHL20D | |
| Es_UY.IBM-1145.le | CUOHL20E | |
| Es_UY.IBM-284.le | CUOHL20F | |
| es_UY.IBM-923.le | CUOHL2FC | |
| Es_UY.IBM-923.le | CUOHL2FC | |
| Es_UY.IBM-924.le | CUOHL21A | |
| es_UY.ISO8859-1.le | CUOHL2FB | |
| Es_UY.ISO8859-1.le | CUOHL2FB | |
| Es_VE.IBM-1047.le | CUOHL21B | |
| Es_VE.IBM-1145.le | CUOHL21C | |
| Es_VE.IBM-284.le | CUOHL21D | |
| es_VE.IBM-923.le | CUOHL2FE | |
| Es_VE.IBM-923.le | CUOHL2FE | |
| Es_VE.IBM-924.le | CUOHL21E | |
| es_VE.ISO8859-1.le | CUOHL2FD | |
| Es_VE.ISO8859-1.le | CUOHL2FD | |
| Es_VEO.IBM-1047.le | CUOHL21F | |
| Es_VEO.IBM-1145.le | CUOHL220 | |
| Es_VEO.IBM-284.le | CUOHL221 | |
| es_VEO.IBM-923.le | CUOHL300 | |
| Es_VEO.IBM-923.le | CUOHL300 | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| Es_VEO.IBM-924.le | CUOHL222 | |
| es_VEO.ISO8859-1.le | CUOHL2FF | |
| Es_VEO.ISO8859-1.le | CUOHL2FF | |
| Et_EE.IBM-1122.le | CUOHL223 | |
| Et_EE.IBM-1157.le | CUOHL225 | |
| Et_EE.IBM-1157@euro.le | CUOHL224 | |
| Et_EE.IBM-1157@preeuro.le | CUOHL225 | |
| Et_EE.le | CUOHL223 | 01122 |
| et_EE.UTF-8.le | CUOHL301 | |
| Et_EE.UTF-8.le | CUOHL301 | |
| et_EE.UTF-8@euro.le | CUOHL302 | |
| Et_EE.UTF-8@euro.le | CUOHL302 | |
| et_EE.UTF-8@preeuro.le | CUOHL301 | |
| Et_EE.UTF-8@preeuro.le | CUOHL301 | |
| eu_ES.UTF-8.le | CUOHL303 | 01208 |
| Eu_ES.UTF-8.le | CUOHL303 | 01208 |
| Fi_FI.IBM-1047.le | CUOHL226 | |
| Fi_FI.IBM-1143.le | CUOHL227 | |
| Fi_FI.IBM-1143@euro.le | CUOHL227 | |
| Fi_FI.IBM-1143@preeuro.le | CUOHL228 | |
| Fi_FI.IBM-278.le | CUOHL226 | |
| Fi_FI.IBM-924.le | CUOHL229 | |
| Fi_FI.IBM-924@euro.le | CUOHL229 | |
| Fi_FI.IBM-924@preeuro.le | CUOHL22A | |
| fi_FI.ISO8859-1.le | CUOHL304 | |
| Fi_FI.ISO8859-1.le | CUOHL304 | |
| Fi_FI.le | CUOHL226 | 01047 |
| fi_FI.UTF-8.le | CUOHL305 | |
| Fi_FI.UTF-8.le | CUOHL305 | |
| fi_FI.UTF-8@euro.le | CUOHL305 | |
| Fi_FI.UTF-8@euro.le | CUOHL305 | |
| fi_FI.UTF-8@preeuro.le | CUOHL306 | |
| Fi_FI.UTF-8@preeuro.le | CUOHL306 | |
| Fr_BE.IBM-1047.le | CUOHL22B | |
| Fr_BE.IBM-1148.le | CUOHL22C | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|---------------|-----------------------------------|
| Fr_BE.IBM-1148@euro.le | CUOHL22C | |
| Fr_BE.IBM-1148@preeuro.le | CUOHL22D | |
| Fr_BE.IBM-500.le | CUOHL22B | |
| Fr_BE.IBM-924.le | CUOHL22E | |
| Fr_BE.IBM-924@euro.le | CUOHL22E | |
| Fr_BE.IBM-924@preeuro.le | CUOHL22F | |
| fr_BE.ISO8859-1.le | CUOHL307 | |
| Fr_BE.ISO8859-1.le | CUOHL307 | |
| Fr_BE.le | CUOHL22B | 01047 |
| fr_BE.UTF-8.le | CUOHL308 | |
| Fr_BE.UTF-8.le | CUOHL308 | |
| fr_BE.UTF-8@euro.le | CUOHL308 | |
| Fr_BE.UTF-8@euro.le | CUOHL308 | |
| fr_BE.UTF-8@preeuro.le | CUOHL309 | |
| Fr_BE.UTF-8@preeuro.le | CUOHL309 | |
| Fr_CA.IBM-037.le | CUOHL230 | |
| Fr_CA.IBM-1047.le | CUOHL230 | |
| Fr_CA.IBM-1140.le | CUOHL231 | |
| Fr_CA.IBM-1140@euro.le | CUOHL232 | |
| Fr_CA.IBM-500.le | CUOHL233 | |
| fr_CA.ISO8859-1.le | CUOHL30A | |
| Fr_CA.ISO8859-1.le | CUOHL30A | |
| Fr_CA.le | CUOHL230 | 01047 |
| fr_CA.UTF-8.le | CUOHL30B | |
| Fr_CA.UTF-8.le | CUOHL30B | |
| Fr_CH.IBM-1047.le | CUOHL234 | |
| Fr_CH.IBM-1148.le | CUOHL235 | |
| Fr_CH.IBM-1148@euro.le | CUOHL236 | |
| Fr_CH.IBM-500.le | CUOHL234 | |
| fr_CH.ISO8859-1.le | CUOHL30C | |
| Fr_CH.ISO8859-1.le | CUOHL30C | |
| Fr_CH.le | CUOHL234 | 01047 |
| fr_CH.UTF-8.le | CUOHL30D | |
| Fr_CH.UTF-8.le | CUOHL30D | |
| Fr_FR.IBM-1047.le | CUOHL237 | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| Fr_FR.IBM-1147.le | CUOHL238 | |
| Fr_FR.IBM-1147@euro.le | CUOHL238 | |
| Fr_FR.IBM-1147@preeuro.le | CUOHL239 | |
| Fr_FR.IBM-297.le | CUOHL237 | |
| Fr_FR.IBM-924.le | CUOHL23A | |
| Fr_FR.IBM-924@euro.le | CUOHL23A | |
| Fr_FR.IBM-924@preeuro.le | CUOHL23B | |
| fr_FR.ISO8859-1.le | CUOHL30E | |
| Fr_FR.ISO8859-1.le | CUOHL30E | |
| Fr_FR.le | CUOHL237 | 01047 |
| fr_FR.UTF-8.le | CUOHL30F | |
| Fr_FR.UTF-8.le | CUOHL30F | |
| fr_FR.UTF-8@euro.le | CUOHL30F | |
| Fr_FR.UTF-8@euro.le | CUOHL30F | |
| fr_FR.UTF-8@preeuro.le | CUOHL310 | |
| Fr_FR.UTF-8@preeuro.le | CUOHL310 | |
| Fr_LU.IBM-924.le | CUOHL23C | |
| Fr_LU.IBM-924@euro.le | CUOHL23C | |
| Fr_LU.IBM-924@preeuro.le | CUOHL23D | |
| fr_LU.UTF-8.le | CUOHL311 | |
| Fr_LU.UTF-8.le | CUOHL311 | |
| fr_LU.UTF-8@euro.le | CUOHL311 | |
| Fr_LU.UTF-8@euro.le | CUOHL311 | |
| fr_LU.UTF-8@preeuro.le | CUOHL312 | |
| Fr_LU.UTF-8@preeuro.le | CUOHL312 | |
| gl_ES.UTF-8.le | CUOHL313 | 01208 |
| Gl_ES.UTF-8.le | CUOHL313 | 01208 |
| gu_IN.UTF-8.le | CUOHL314 | 01208 |
| Gu_IN.UTF-8.le | CUOHL314 | 01208 |
| he_IL.ISO8859-8.le | CUOHL315 | |
| He_IL.ISO8859-8.le | CUOHL315 | |
| he_IL.UTF-8.le | CUOHL316 | |
| He_IL.UTF-8.le | CUOHL316 | |
| hi_IN.UTF-8.le | CUOHL317 | 01208 |
| Hi_IN.UTF-8.le | CUOHL317 | 01208 |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|---------------|-----------------------------------|
| Hr_HR.IBM-1153.le | CUOHL23E | |
| Hr_HR.IBM-1165.le | CUOHL23F | |
| Hr_HR.IBM-870.le | CUOHL240 | |
| hr_HR.ISO8859-2.le | CUOHL318 | |
| Hr_HR.ISO8859-2.le | CUOHL318 | |
| Hr_HR.le | CUOHL240 | 00870 |
| hr_HR.UTF-8.le | CUOHL319 | |
| Hr_HR.UTF-8.le | CUOHL319 | |
| Hu_HU.IBM-1153.le | CUOHL242 | |
| Hu_HU.IBM-1153@euro.le | CUOHL241 | |
| Hu_HU.IBM-1153@preeuro.le | CUOHL242 | |
| Hu_HU.IBM-1165.le | CUOHL244 | |
| Hu_HU.IBM-1165@euro.le | CUOHL243 | |
| Hu_HU.IBM-1165@preeuro.le | CUOHL244 | |
| Hu_HU.IBM-870.le | CUOHL245 | |
| hu_HU.ISO8859-2.le | CUOHL31A | |
| Hu_HU.ISO8859-2.le | CUOHL31A | |
| Hu_HU.le | CUOHL245 | 00870 |
| hu_HU.UTF-8.le | CUOHL31B | |
| Hu_HU.UTF-8.le | CUOHL31B | |
| hu_HU.UTF-8@euro.le | CUOHL31C | |
| Hu_HU.UTF-8@euro.le | CUOHL31C | |
| hu_HU.UTF-8@preeuro.le | CUOHL31B | |
| Hu_HU.UTF-8@preeuro.le | CUOHL31B | |
| hy_AM.UTF-8.le | CUOHL31D | 01208 |
| Hy_AM.UTF-8.le | CUOHL31D | 01208 |
| Id_ID.IBM-1047.le | CUOHL246 | |
| id_ID.ISO8859-1.le | CUOHL31E | |
| Id_ID.ISO8859-1.le | CUOHL31E | |
| Is_IS.IBM-1047.le | CUOHL247 | |
| Is_IS.IBM-1149.le | CUOHL248 | |
| Is_IS.IBM-1149@euro.le | CUOHL249 | |
| Is_IS.IBM-871.le | CUOHL247 | |
| Is_IS.le | CUOHL247 | 01047 |
| is_IS.UTF-8.le | CUOHL31F | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| Is_IS.UTF-8.le | CUOHL31F | |
| It_CH.IBM-1047.le | CUOHL24A | |
| It_CH.IBM-1148.le | CUOHL24B | |
| It_CH.IBM-500.le | CUOHL24C | |
| it_CH.IBM-923.le | CUOHL321 | |
| It_CH.IBM-923.le | CUOHL321 | |
| It_CH.IBM-924.le | CUOHL24D | |
| it_CH.ISO8859-1.le | CUOHL320 | |
| It_CH.ISO8859-1.le | CUOHL320 | |
| It_IT.IBM-1047.le | CUOHL24E | |
| It_IT.IBM-1144.le | CUOHL24F | |
| It_IT.IBM-1144@euro.le | CUOHL24F | |
| It_IT.IBM-1144@preeuro.le | CUOHL250 | |
| It_IT.IBM-280.le | CUOHL24E | |
| It_IT.IBM-924.le | CUOHL251 | |
| It_IT.IBM-924@euro.le | CUOHL251 | |
| It_IT.IBM-924@preeuro.le | CUOHL252 | |
| it_IT.ISO8859-1.le | CUOHL322 | |
| It_IT.ISO8859-1.le | CUOHL322 | |
| It_IT.le | CUOHL24E | 01047 |
| it_IT.UTF-8.le | CUOHL323 | |
| It_IT.UTF-8.le | CUOHL323 | |
| it_IT.UTF-8@euro.le | CUOHL323 | |
| It_IT.UTF-8@euro.le | CUOHL323 | |
| it_IT.UTF-8@preeuro.le | CUOHL324 | |
| It_IT.UTF-8@preeuro.le | CUOHL324 | |
| Iw_IL.IBM-12712.le | CUOHL253 | |
| Iw_IL.IBM-424.le | CUOHL254 | |
| Iw_IL.IBM12712.le | CUOHL253 | |
| iw_IL.ISO8859-8.le | CUOHL325 | |
| Iw_IL.ISO8859-8.le | CUOHL325 | |
| Iw_IL.le | CUOHL254 | 00424 |
| iw_IL.UTF-8.le | CUOHL316 | |
| Iw_IL.UTF-8.le | CUOHL316 | |
| Ja_JP.IBM-1027.le | CUOHL255 | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|---------------|-----------------------------------|
| Ja_JP.IBM-1390.le | CUOHL256 | |
| Ja_JP.IBM-1399.le | CUOHL257 | |
| Ja_JP.IBM-290.le | CUOHL2AE | |
| Ja_JP.IBM-5123.le | CUOHL258 | |
| Ja_JP.IBM-8482.le | CUOHL259 | |
| Ja_JP.IBM-930.le | CUOHL2AF | |
| Ja_JP.IBM-939.le | CUOHL25A | |
| ja_JP.IBM-943.le | CUOHL326 | |
| Ja_JP.IBM-943.le | CUOHL326 | |
| Ja_JP.le | CUOHL25A | 00939 |
| ja_JP.UTF-8.le | CUOHL327 | |
| Ja_JP.UTF-8.le | CUOHL327 | |
| ka_GE.UTF-8.le | CUOHL328 | 01208 |
| Ka_GE.UTF-8.le | CUOHL328 | 01208 |
| kk_KZ.UTF-8.le | CUOHL329 | 01208 |
| Kk_KZ.UTF-8.le | CUOHL329 | 01208 |
| kn_IN.UTF-8.le | CUOHL32A | 01208 |
| Kn_IN.UTF-8.le | CUOHL32A | 01208 |
| ko_KR.IBM-eucKR.le | CUOHL32B | |
| Ko_KR.IBM-eucKR.le | CUOHL32B | |
| Ko_KR.IBM-1364.le | CUOHL25B | |
| Ko_KR.IBM-933.le | CUOHL25C | |
| Ko_KR.le | CUOHL25C | 00933 |
| ko_KR.UTF-8.le | CUOHL32C | |
| Ko_KR.UTF-8.le | CUOHL32C | |
| Lt_LT.IBM-1112.le | CUOHL25D | |
| Lt_LT.IBM-1156.le | CUOHL25F | |
| Lt_LT.IBM-1156@euro.le | CUOHL25E | |
| Lt_LT.IBM-1156@preeuro.le | CUOHL25F | |
| Lt_LT.le | CUOHL25D | 01112 |
| lt_LT.UTF-8.le | CUOHL32D | |
| Lt_LT.UTF-8.le | CUOHL32D | |
| lt_LT.UTF-8@euro.le | CUOHL32E | |
| Lt_LT.UTF-8@euro.le | CUOHL32E | |
| lt_LT.UTF-8@preeuro.le | CUOHL32D | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| Lt_LT.UTF-8@preeuro.le | CUOHL32D | |
| Lv_LV.IBM-1112.le | CUOHL260 | |
| Lv_LV.IBM-1156.le | CUOHL262 | |
| Lv_LV.IBM-1156@euro.le | UOHL261 | |
| Lv_LV.IBM-1156@preeuro.le | CUOHL262 | |
| lv_LV.IBM-901.le | CUOHL32F | |
| Lv_LV.IBM-901.le | CUOHL32F | |
| lv_LV.IBM-921.le | CUOHL330 | |
| Lv_LV.IBM-921.le | CUOHL330 | |
| lv_LV.UTF-8.le | CUOHL331 | |
| Lv_LV.UTF-8.le | CUOHL331 | |
| lv_LV.UTF-8@euro.le | CUOHL332 | |
| Lv_LV.UTF-8@euro.le | CUOHL332 | |
| lv_LV.UTF-8@preeuro.le | CUOHL331 | |
| Lv_LV.UTF-8@preeuro.le | CUOHL331 | |
| Mk_MK.IBM-1025.le | CUOHL263 | |
| Mk_MK.IBM-1154.le | CUOHL264 | |
| Mk_MK.le | CUOHL263 | 01025 |
| mr_IN.UTF-8.le | CUOHL333 | 01208 |
| Mr_IN.UTF-8.le | CUOHL333 | 01208 |
| Ms_MY.IBM-1047.le | CUOHL265 | |
| ms_MY.ISO8859-1.le | CUOHL334 | |
| Ms_MY.ISO8859-1.le | CUOHL334 | |
| mt_MT.UTF-8.le | CUOHL336 | 01208 |
| Mt_MT.UTF-8.le | CUOHL336 | 01208 |
| mt_MT.UTF-8@euro.le | CUOHL336 | 01208 |
| Mt_MT.UTF-8@euro.le | CUOHL336 | 01208 |
| mt_MT.UTF-8@preeuro.le | CUOHL335 | 01208 |
| Mt_MT.UTF-8@preeuro.le | CUOHL335 | 01208 |
| nb_NO.UTF-8.le | CUOHL337 | 01208 |
| Nb_NO.UTF-8.le | CUOHL337 | 01208 |
| Nl_BE.IBM-1047.le | CUOHL266 | |
| Nl_BE.IBM-1148.le | CUOHL267 | |
| Nl_BE.IBM-1148@euro.le | CUOHL267 | |
| Nl_BE.IBM-1148@preeuro.le | CUOHL268 | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|---------------|-----------------------------------|
| Nl_BE.IBM-500.le | CUOHL266 | |
| Nl_BE.IBM-924.le | CUOHL269 | |
| Nl_BE.IBM-924@euro.le | CUOHL269 | |
| Nl_BE.IBM-924@preeuro.le | CUOHL26A | |
| Nl_BE.le | CUOHL266 | 01047 |
| nl_BE.UTF-8.le | CUOHL338 | |
| Nl_BE.UTF-8.le | CUOHL338 | |
| nl_BE.UTF-8@euro.le | CUOHL338 | |
| Nl_BE.UTF-8@euro.le | CUOHL338 | |
| nl_BE.UTF-8@preeuro.le | CUOHL339 | |
| Nl_BE.UTF-8@preeuro.le | CUOHL339 | |
| Nl_NL.IBM-037.le | CUOHL26B | |
| Nl_NL.IBM-1047.le | CUOHL26B | |
| Nl_NL.IBM-1140.le | CUOHL26C | |
| Nl_NL.IBM-1140@euro.le | CUOHL26C | |
| Nl_NL.IBM-1140@preeuro.le | CUOHL26D | |
| Nl_NL.IBM-924.le | CUOHL26E | |
| Nl_NL.IBM-924@euro.le | CUOHL26E | |
| Nl_NL.IBM-924@preeuro.le | CUOHL26F | |
| nl_NL.ISO8859-1.le | CUOHL33A | |
| Nl_NL.ISO8859-1.le | CUOHL33A | |
| Nl_NL.le | CUOHL26B | 01047 |
| nl_NL.UTF-8.le | CUOHL33B | |
| Nl_NL.UTF-8.le | CUOHL33B | |
| nl_NL.UTF-8@euro.le | CUOHL33B | |
| Nl_NL.UTF-8@euro.le | CUOHL33B | |
| nl_NL.UTF-8@preeuro.le | CUOHL33C | |
| Nl_NL.UTF-8@preeuro.le | CUOHL33C | |
| nn_NO.UTF-8.le | CUOHL33D | 01208 |
| Nn_NO.UTF-8.le | CUOHL33D | 01208 |
| No_NO.IBM-1047.le | CUOHL270 | |
| No_NO.IBM-1142.le | CUOHL271 | |
| No_NO.IBM-1142@euro.le | CUOHL272 | |
| No_NO.IBM-277.le | CUOHL270 | |
| no_NO.ISO8859-1.le | CUOHL33E | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| No_NO.ISO8859-1.le | CUOHL33E | |
| No_NO.le | CUOHL270 | 01047 |
| no_NO.UTF-8.le | CUOHL33F | |
| No_NO.UTF-8.le | CUOHL33F | |
| pa_IN.UTF-8.le | CUOHL340 | 01208 |
| Pa_IN.UTF-8.le | CUOHL340 | 01208 |
| Pl_PL.IBM-1153.le | CUOHL274 | |
| Pl_PL.IBM-1153@euro.le | CUOHL273 | |
| Pl_PL.IBM-1153@preeuro.le | CUOHL274 | |
| Pl_PL.IBM-1165.le | CUOHL276 | |
| Pl_PL.IBM-1165@euro.le | CUOHL275 | |
| Pl_PL.IBM-1165@preeuro.le | CUOHL276 | |
| Pl_PL.IBM-870.le | CUOHL277 | |
| pł_PL.ISO8859-2.le | CUOHL341 | |
| Pl_PL.ISO8859-2.le | CUOHL341 | |
| Pl_PL.le | CUOHL277 | 00870 |
| pł_PL.UTF-8.le | CUOHL342 | |
| Pl_PL.UTF-8.le | CUOHL342 | |
| pł_PL.UTF-8@euro.le | CUOHL343 | |
| Pl_PL.UTF-8@euro.le | CUOHL343 | |
| pł_PL.UTF-8@preeuro.le | CUOHL342 | |
| Pl_PL.UTF-8@preeuro.le | CUOHL342 | |
| POSIX.le | CUOHL278 | |
| Pt_BR.IBM-037.le | CUOHL279 | |
| Pt_BR.IBM-1047.le | CUOHL279 | |
| Pt_BR.IBM-1140.le | CUOHL27A | |
| Pt_BR.IBM-1140@euro.le | CUOHL27B | |
| pt_BR.ISO8859-1.le | CUOHL344 | |
| Pt_BR.ISO8859-1.le | CUOHL344 | |
| Pt_BR.le | CUOHL279 | 01047 |
| pt_BR.UTF-8.le | CUOHL345 | |
| Pt_BR.UTF-8.le | CUOHL345 | |
| Pt_PT.IBM-037.le | CUOHL27C | |
| Pt_PT.IBM-1047.le | CUOHL27C | |
| Pt_PT.IBM-1140.le | CUOHL27D | |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|---------------|-----------------------------------|
| Pt_PT.IBM-1140@euro.le | CUOHL27D | |
| Pt_PT.IBM-1140@preeuro.le | CUOHL27E | |
| Pt_PT.IBM-924.le | CUOHL27F | |
| Pt_PT.IBM-924@euro.le | CUOHL27F | |
| Pt_PT.IBM-924@preeuro.le | CUOHL280 | |
| pt_PT.ISO8859-1.le | CUOHL346 | |
| Pt_PT.ISO8859-1.le | CUOHL346 | |
| Pt_PT.le | CUOHL27C | 01047 |
| pt_PT.UTF-8.le | CUOHL347 | |
| Pt_PT.UTF-8.le | CUOHL347 | |
| pt_PT.UTF-8@euro.le | CUOHL347 | |
| Pt_PT.UTF-8@euro.le | CUOHL347 | |
| pt_PT.UTF-8@preeuro.le | CUOHL348 | |
| Pt_PT.UTF-8@preeuro.le | CUOHL348 | |
| Ro_RO.IBM-1153.le | CUOHL281 | |
| Ro_RO.IBM-1153@euro.le | CUOHL282 | |
| Ro_RO.IBM-1153@preeuro.le | CUOHL281 | |
| Ro_RO.IBM-1165.le | CUOHL283 | |
| Ro_RO.IBM-1165@euro.le | CUOHL284 | |
| Ro_RO.IBM-1165@preeuro.le | CUOHL283 | |
| Ro_RO.IBM-870.le | CUOHL285 | |
| ro_RO.ISO8859-2.le | CUOHL349 | |
| Ro_RO.ISO8859-2.le | CUOHL349 | |
| Ro_RO.le | CUOHL285 | 00870 |
| ro_RO.UTF-8.le | CUOHL34A | |
| Ro_RO.UTF-8.le | CUOHL34A | |
| ro_RO.UTF-8@euro.le | CUOHL34B | |
| Ro_RO.UTF-8@euro.le | CUOHL34B | |
| ro_RO.UTF-8@preeuro.le | CUOHL34A | |
| Ro_RO.UTF-8@preeuro.le | CUOHL34A | |
| Ru_RU.IBM-1025.le | CUOHL286 | |
| Ru_RU.IBM-1154.le | CUOHL287 | |
| ru_RU.ISO8859-5.le | CUOHL34C | |
| Ru_RU.ISO8859-5.le | CUOHL34C | |
| Ru_RU.le | CUOHL286 | 01025 |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|----------|----------------------------|
| ru_RU.UTF-8.le | CUOHL34D | |
| Ru_RU.UTF-8.le | CUOHL34D | |
| SAA.le | CUOHL288 | |
| sh_CS.UTF-8.le | CUOHL34E | 01208 |
| Sh_CS.UTF-8.le | CUOHL34E | 01208 |
| Sh_SP.IBM-1153.le | CUOHL289 | |
| Sh_SP.IBM-1165.le | CUOHL28A | |
| Sh_SP.IBM-870.le | CUOHL28B | |
| Sh_SP.le | CUOHL28B | 00870 |
| Sk_SK.IBM-1153.le | CUOHL28C | |
| Sk_SK.IBM-1153@euro.le | CUOHL28C | |
| Sk_SK.IBM-1153@preeuro.le | CUOHL28D | |
| Sk_SK.IBM-1165.le | CUOHL28E | |
| Sk_SK.IBM-1165@euro.le | CUOHL28E | |
| Sk_SK.IBM-1165@preeuro.le | CUOHL28F | |
| Sk_SK.IBM-870.le | CUOHL290 | |
| sk_SK.ISO8859-2.le | CUOHL34F | |
| Sk_SK.ISO8859-2.le | CUOHL34F | |
| Sk_SK.le | CUOHL290 | 00870 |
| sk_SK.UTF-8.le | CUOHL351 | |
| Sk_SK.UTF-8.le | CUOHL351 | |
| sk_SK.UTF-8@euro.le | CUOHL351 | |
| Sk_SK.UTF-8@euro.le | CUOHL351 | |
| sk_SK.UTF-8@preeuro.le | CUOHL350 | |
| Sk_SK.UTF-8@preeuro.le | CUOHL350 | |
| Sl_SI.IBM-1153.le | CUOHL291 | |
| Sl_SI.IBM-1153@euro.le | CUOHL291 | |
| Sl_SI.IBM-1153@preeuro.le | CUOHL292 | |
| Sl_SI.IBM-1165.le | CUOHL293 | |
| Sl_SI.IBM-1165@euro.le | CUOHL293 | |
| Sl_SI.IBM-1165@preeuro.le | CUOHL294 | |
| Sl_SI.IBM-870.le | CUOHL295 | |
| sl_SI.ISO8859-2.le | CUOHL352 | |
| Sl_SI.ISO8859-2.le | CUOHL352 | |
| Sl_SI.le | CUOHL295 | 00870 |

| Locale name | Member | Default CCSID (if defined) |
|---------------------------|---------------|-----------------------------------|
| sl_SI.UTF-8.le | CUOHL353 | |
| Sl_SI.UTF-8.le | CUOHL353 | |
| sl_SI.UTF-8@euro.le | CUOHL354 | |
| Sl_SI.UTF-8@euro.le | CUOHL354 | |
| sl_SI.UTF-8@preeuro.le | CUOHL353 | |
| Sl_SI.UTF-8@preeuro.le | CUOHL353 | |
| Sq_AL.IBM-1047.le | CUOHL296 | |
| Sq_AL.IBM-1148.le | CUOHL297 | |
| Sq_AL.IBM-1148@euro.le | CUOHL298 | |
| Sq_AL.IBM-500.le | CUOHL296 | |
| Sq_AL.le | CUOHL296 | 01047 |
| sq_AL.UTF-8.le | CUOHL355 | |
| Sq_AL.UTF-8.le | CUOHL355 | |
| sr_CS.UTF-8.le | CUOHL356 | 01208 |
| Sr_CS.UTF-8.le | CUOHL356 | 01208 |
| sr_RS.UTF-8.le | CUOHL357 | 01208 |
| Sr_RS.UTF-8.le | CUOHL357 | 01208 |
| Sr_SP.IBM-1025.le | CUOHL299 | |
| Sr_SP.IBM-1154.le | CUOHL29A | |
| Sr_SP.le | CUOHL299 | 01025 |
| Sv_SE.IBM-1047.le | CUOHL29B | |
| Sv_SE.IBM-1143.le | CUOHL29D | |
| Sv_SE.IBM-1143@euro.le | CUOHL29C | |
| Sv_SE.IBM-1143@preeuro.le | CUOHL29D | |
| Sv_SE.IBM-278.le | CUOHL29B | |
| Sv_SE.IBM-924.le | CUOHL29F | |
| Sv_SE.IBM-924@euro.le | CUOHL29E | |
| Sv_SE.IBM-924@preeuro.le | CUOHL29F | |
| sv_SE.ISO8859-1.le | CUOHL358 | |
| Sv_SE.ISO8859-1.le | CUOHL358 | |
| Sv_SE.le | CUOHL29B | 01047 |
| sv_SE.UTF-8.le | CUOHL359 | |
| Sv_SE.UTF-8.le | CUOHL359 | |
| sv_SE.UTF-8@euro.le | CUOHL35A | |
| Sv_SE.UTF-8@euro.le | CUOHL35A | |

| Locale name | Member | Default CCSID (if defined) |
|------------------------|----------|----------------------------|
| sv_SE.UTF-8@preeuro.le | CUOHL359 | |
| Sv_SE.UTF-8@preeuro.le | CUOHL359 | |
| sw_KE.UTF-8.le | CUOHL35B | 01208 |
| Sw_KE.UTF-8.le | CUOHL35B | 01208 |
| sw_TZ.UTF-8.le | CUOHL35C | 01208 |
| Sw_TZ.UTF-8.le | CUOHL35C | 01208 |
| ta_IN.UTF-8.le | CUOHL35D | 01208 |
| Ta_IN.UTF-8.le | CUOHL35D | 01208 |
| te_IN.UTF-8.le | CUOHL35E | 01208 |
| Te_IN.UTF-8.le | CUOHL35E | 01208 |
| Th_TH.IBM-1160.le | CUOHL2A0 | |
| th_TH.IBM-838.le | CUOHL371 | |
| th_TH.le | CUOHL371 | 00838 |
| th_TH.TIS-620.le | CUOHL35F | |
| Th_TH.TIS-620.le | CUOHL35F | |
| th_TH.UTF-8.le | CUOHL360 | |
| Th_TH.UTF-8.le | CUOHL360 | |
| Tr_TR.IBM-1026.le | CUOHL2B0 | |
| Tr_TR.IBM-1155.le | CUOHL2A1 | |
| tr_TR.ISO8859-9.le | CUOHL361 | |
| Tr_TR.ISO8859-9.le | CUOHL361 | |
| tr_TR.UTF-8.le | CUOHL362 | |
| Tr_TR.UTF-8.le | CUOHL362 | |
| Tr_TRO.IBM-1023.le | CUOHL2A2 | |
| Tr_TRO.IBM-1155.le | CUOHL2A3 | |
| tr_TRO.ISO8859-9.le | CUOHL364 | |
| Tr_TRO.ISO8859-9.le | CUOHL364 | |
| tr_TRO.UTF-8.le | CUOHL363 | |
| Tr_TRO.UTF-8.le | CUOHL363 | |
| Uk_UA.IBM-1123.le | CUOHL2A4 | |
| uk_UA.IBM-1124.le | CUOHL365 | |
| Uk_UA.IBM-1124.le | CUOHL365 | |
| Uk_UA.IBM-1158.le | CUOHL2A5 | |
| vi_VN.UTF-8.le | CUOHL366 | 01208 |
| Vi_VN.UTF-8.le | CUOHL366 | 01208 |

| Locale name | Member | Default CCSID (if defined) |
|--------------------|---------------|-----------------------------------|
| zh_CN.IBM-eucCN.le | CUOHL367 | |
| Zh_CN.IBM-eucCN.le | CUOHL367 | |
| Zh_CN.IBM-1388.le | CUOHL2A6 | |
| Zh_CN.IBM-935.le | CUOHL2A7 | |
| Zh_CN.le | CUOHL2A7 | 00935 |
| zh_CN.UTF-8.le | CUOHL368 | |
| Zh_CN.UTF-8.le | CUOHL368 | |
| zh_HK.UTF-8.le | CUOHL36B | 01208 |
| Zh_HK.UTF-8.le | CUOHL36B | 01208 |
| Zh_HKS.IBM-1388.le | CUOHL2A9 | |
| Zh_HKS.IBM-935.le | CUOHL2A8 | |
| zh_HKS.UTF-8.le | CUOHL369 | 01208 |
| Zh_HKS.UTF-8.le | CUOHL369 | 01208 |
| zh_HKT.UTF-8.le | CUOHL36A | 01208 |
| Zh_HKT.UTF-8.le | CUOHL36A | 01208 |
| zh_SG.UTF-8.le | CUOHL36D | 01208 |
| Zh_SG.UTF-8.le | CUOHL36D | 01208 |
| Zh_SGS.IBM-1388.le | CUOHL2AB | |
| Zh_SGS.IBM-935.le | CUOHL2AA | |
| zh_SGS.UTF-8.le | CUOHL36C | 01208 |
| Zh_SGS.UTF-8.le | CUOHL36C | 01208 |
| zh_TW.BIG5.le | CUOHL36E | |
| Zh_TW.BIG5.le | CUOHL36E | |
| Zh_TW.IBM-1371.le | CUOHL2AD | |
| Zh_TW.IBM-937.le | CUOHL2AC | |
| Zh_TW.le | CUOHL2AC | 00937 |
| zh_TW.UTF-8.le | CUOHL36F | |
| Zh_TW.UTF-8.le | CUOHL36F | |
| zu_ZA.UTF-8.le | CUOHL370 | 01208 |
| Zu_ZA.UTF-8.le | CUOHL370 | 01208 |

Note: The default character mappings listed in [Table 88 on page 557](#) correspond to the current default character mappings provided by the C/C++ Run-time Library when a `setlocale()` call is made without specifying a character mapping.

Appendix G. System control offsets

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 89. Offsets for 31-bit callers. | |
|---------------------------------------|----------------|
| Interface description | Decimal offset |
| Character conversion | 172 |
| Case conversion | 180 |
| Normalization | 212 |
| Collation | 228 |
| String Preparation | 152 |
| Bidi (See note) | 136 |

Note: IBM does not intend to enhance the bidi transformation service. Instead, it is recommended that you use the character conversion 'extended bidi support' for all new development and for the highest level of bidi support.

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 90. Offsets for 64-bit callers. | |
|---------------------------------------|----------------|
| Interface description | Decimal offset |
| Character conversion | 200 |
| Case conversion | 192 |

| <i>Table 90. Offsets for 64-bit callers. (continued)</i> | |
|--|-----------------------|
| Interface description | Decimal offset |
| Normalization | 216 |
| Collation | 232 |
| String Preparation | 156 |
| Bidi (See note) | 140 |

Note: IBM does not intend to enhance the bidi transformation service. Instead, it is recommended that you use the character conversion 'extended bidi support' for all new development and for the highest level of bidi support.

Appendix H. Unicode return and reason codes

This topic includes z/OS support for Unicode return and reason codes.

Return code meanings

| <i>Table 91. Classification of return codes</i> | | |
|---|-----------------|---|
| 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.

| <i>Table 92. Return and reason codes from z/OS Unicode Services</i> | | | |
|---|-------------------------|--|------------|
| 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 |

| Table 92. Return and reason codes from z/OS Unicode Services (continued) | | | |
|--|-------------------------|---|------------|
| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
| 4 | 3 | <p>Name: CUN_RS_INV_HANDLE_SET</p> <p>Meaning: Conversion is terminated. The handle is invalid because a SET UNI command is in process and will change the conversion environment.</p> <p>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.</p> | Conversion |
| 4 | 4 | <p>Name: CUN_RS_NO_HANDLE</p> <p>Meaning: Conversion is terminated. No handle can be obtained because a SET UNI command is in process and will change the conversion environment.</p> <p>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.</p> | Conversion |
| 4 | 6 | <p>Name: CUN_RS_SUB_ACT_TERM</p> <p>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.</p> <p>Action:</p> <ol style="list-style-type: none"> 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 | <p>Name: CUN_RS_MBC_INCOMPLETE</p> <p>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.</p> <p>Action: Check whether the input string is correct. Make sure that the missing bytes are in the source string.</p> | Conversion |
| 4 | 8 | <p>Name: CUN_RS_CONTINUATION</p> <p>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.</p> <p>Action: Next call to casing service needs to start with the uncased character of this call as the first source character.</p> | Case |
| 4 | 9 | <p>Name: CUN_RS_STAGE2_FAIL</p> <p>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.</p> <p>Action: Check whether the input string and the parameter settings used are reasonable.</p> | Conversion |

| Table 92. Return and reason codes from z/OS Unicode Services (continued) | | | |
|--|-------------------------|---|---------------|
| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
| 4 | 0A | <p>Name: CUN_RS_WRK_EXH</p> <p>Meaning: The work buffer was exhausted before all characters in the target buffer could be processed.</p> <p>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.</p> | Normalization |
| 4 | 0B | <p>Name: CUN_RS_SOURCE_LEN_ZERO</p> <p>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.</p> <p>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.</p> | Collation |
| 4 | 0C | <p>Name: CUN_RS_MAL_CHAR_ACT_TERM</p> <p>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”.</p> <p>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.</p> | Conversion |
| 4 | 0D | <p>Name: CUN_RS_INVALID_COLL_DATA_VER</p> <p>Meaning: The specified Collation version is already loaded into the Unicode DataSpace.</p> <p>Action: Check whether the specified collation version is correct and recall the service.</p> | Collation |
| 4 | 0E | <p>Name: CUN_RS_INVALID_ALTERNATE_VALUE</p> <p>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.</p> <p>Action: Call the service again with a valid alternate value:</p> <ul style="list-style-type: none"> • ALTERNATE_NON_IGNOREABLE • ALTERNATE_SHIFTED | Collation |
| 4 | 0F | <p>Name: CUN_RS_INVALID_NORMALIZATION_VALUE</p> <p>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.</p> <p>Action: Call the service again with a valid normalization value:</p> <ul style="list-style-type: none"> • NORMALIZATION_OFF • NORMALIZATION_ON | Collation |

Table 92. Return and reason codes from z/OS Unicode Services (continued)

| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------------------|-------------------------|--|--|--------|---------------|--|----------------------|----------|----------|--|-----------------|----------|----------|--|---------------|----------|----------|--|-----------|----------|----------|--|------|----------|----------|--|------------|----------|----------|--|--------------------------------|----------|----------|--|------------|
| 4 | 10 | <p>Name: CUN_RS_LOCALES_AND_UCR_ARE_EXCLUSIVE</p> <p>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.</p> <p>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.</p> | Collation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 1 | <p>Name: CUN_RS_PARM_VER</p> <p>Meaning: Wrong version of the parameter area used.</p> <p>Action: Use the correct parameter area version constant provided in the following interface definition file.</p> <table><tr><th>z/OS Unicode service</th><th>31-bit</th></tr><tr><td>64-bit</td><td></td></tr><tr><td>Character conversion</td><td>CUNBCIDF</td></tr><tr><td>CUN4BCID</td><td></td></tr><tr><td>Case Conversion</td><td>CUNBAIDF</td></tr><tr><td>CUN4BAID</td><td></td></tr><tr><td>Normalization</td><td>CUNBNIDF</td></tr><tr><td>CUN4BNID</td><td></td></tr><tr><td>Collation</td><td>CUNB0IDF</td></tr><tr><td>CUN4B0ID</td><td></td></tr><tr><td>BIDI</td><td>CUNBBIDF</td></tr><tr><td>CUN4BBID</td><td></td></tr><tr><td>StringPrep</td><td>CUNBPIDF</td></tr><tr><td>CUN4BPID</td><td></td></tr><tr><td>Conversion information service</td><td>CUNBIIDF</td></tr><tr><td>CUN4BIID</td><td></td></tr></table> <p>When the service is called successfully, CUNBIPRM_Return_Code = 0 and CUNBIPRM_Reason_Code = 0.</p> | z/OS Unicode service | 31-bit | 64-bit | | Character conversion | CUNBCIDF | CUN4BCID | | Case Conversion | CUNBAIDF | CUN4BAID | | Normalization | CUNBNIDF | CUN4BNID | | Collation | CUNB0IDF | CUN4B0ID | | 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 | CUNB0IDF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CUN4B0ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BIDI | CUNBBIDF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CUN4BBID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| StringPrep | CUNBPIDF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CUN4BPID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Conversion information service | CUNBIIDF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CUN4BIID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 2 | <p>Name: CUN_RS_WRK_BUF_SMALL</p> <p>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.</p> <p>Action: Call the service again using a work buffer of larger size.</p> | Conversion, Normalization, Collation, StringPrep | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 3 | <p>Name: CUN_RS_CC SID_NOT_SUPP</p> <p>Meaning: The specified conversion is not supported in the current conversion image.</p> <p>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.</p> | All | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 92. Return and reason codes from z/OS Unicode Services (continued)

| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
|----------------------------|----------------------------|---|------------|
| 8 | 4 | <p>Name: CUN_RS_CASE_NOT_SUPP</p> <p>Meaning: It can be one of the following meanings:</p> <ul style="list-style-type: none"> An unsupported case conversion type was specified.Action: Call the service with the conversion type parameter set to a supported conversion type. An invalid locale name was specified in CUNBAPRM_Locale or CUN4BAPR_Locale (31 and 64-bit respectively).Action : Call the service with a valid locale name (See “Locales supported for case service” on page 554). | Case |
| 8 | 5 | <p>Name: CUN_RS_SUBCODEPAGE</p> <p>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.</p> <p>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.</p> | Conversion |
| 8 | 6 | <p>Name: CUN_RS_TRG_BUF_SMALL</p> <p>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.</p> <p>For CASE, Normalization, StrigPrep, and BIDI Unicode Services, target buffer is not large enough to hold at least one UTF-16 BE character.</p> <p>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.</p> <p>Action: Call the service again using a target buffer of adequate length.</p> | All |

Table 92. Return and reason codes from z/OS Unicode Services (continued)

| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
|----------------------------|----------------------------|--|---|
| 8 | 7 | <p>Name: CUN_RS_DDA_BUF_SMALL</p> <p>It can be either of the following reasons:</p> <ul style="list-style-type: none"> • Meaning: The caller supplied a DDA buffer that is not large enough for the storage required by the conversion services. <p>Action: Call the service again using the required DDA_Buf_Len as described by the following constant:</p> <ul style="list-style-type: none"> – For 31-bit callers: <ul style="list-style-type: none"> - 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 normalization (in interface definition file CUNBNIDF) - CUNBOPRM_DDA_Req for collation (in interface definition file CUNBOIDF) - CUNBIPRM_DDA_Req for information service (in interface definition file CUNBIIDF) - CUNBCPRM_DDA_REQ2 for character conversion if CUNBCPRM_Version is set to CUNBCPRM_VER2 (in interface definition file CUNBCIDF). – For 64-bit callers: <ul style="list-style-type: none"> - 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 normalization (in interface definition file CUN4BNID) - CUN4BOPR_DDA_Req for collation (in interface definition file CUN4BOID) - CUN4BIPR_DDA_Req for information service (in interface definition file CUN4BIID) - CUN4BCPR_DDA_REQ2 for character conversion if CUN4BCPR_Version is set to CUN4BCPR_VER2 (in interface definition file CUN4BCID). • 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. <p>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).</p> | Conversion, Case, Normalization, Collation, Information Service |

Table 92. Return and reason codes from z/OS Unicode Services (continued)

| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
|-------------------------|-------------------------|--|--------------------------------------|
| 8 | 8 | <p>Name: CUN_RS_DDA_MIN_SMALL</p> <p>Meaning: The caller supplied a DDA buffer that is not large enough for the storage needed for the initial call to CUNMNCV, CUN4MNCV, CUNMNORM, CUN4MNOR, CUNMOCOL, or CUN4MCOL.</p> <p>Action: You can take one of the following actions:</p> <ul style="list-style-type: none"> For CUNMNCV 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: <ul style="list-style-type: none"> 31-bit callers: <ul style="list-style-type: none"> CUNBNPRM_DDA_Req for character conversion (in interface definition file CUNBNIDF) CUNBOPRM_DDA_Req for case conversion (in interface definition file CUNBOIDF) 64-bit callers: <ul style="list-style-type: none"> CUN4BNPR_DDA_Req for character conversion (in interface definition file CUN4BNID) CUN4BOPR_DDA_Req for case conversion (in interface definition file CUN4BOID) | Conversion, Normalization, Collation |
| 8 | 9 | <p>Name: CUN_RS_INV_NORM_TYPE</p> <p>Meaning: An unsupported normalization type was specified in normalization parameter area (CUNBOPRM).</p> <p>Action: Call the service again using a valid normalization type: CUNBNPRM_D=1, CUNBNPRM_C=2, CUNBNPRM_KD=3, CUNBNPRM_KC=4.</p> | Normalization |
| 8 | 0A | <p>Name: CUN_RS_INV_COLL_LEVEL</p> <p>Meaning: An unsupported collation level was specified.</p> <p>Action: Use a valid collation level in IDF_CUNBOIDF.</p> | Collation |
| 8 | 0B | <p>Name: CUN_RS_NO_SERV_AVAILABLE</p> <p>Meaning: An unavailable service was called in the active image.</p> <p>Action: Use SET command to load an image with the service available.</p> | Case, Normalization, Collation |
| 8 | 0C | <p>Name: CUN_RS_WRK_EXHAUSTED</p> <p>Meaning: The work buffer was exhausted before all the Unicode characters (source buffers) were represented in collation elements (weights – work buffers).</p> <p>Action: Call the service again with new parameter value in the work buffer.</p> | Collation |
| 8 | 0D | <p>Name: CUN_RS_TARG_EXHAUSTED</p> <p>Meaning: The target buffer was exhausted before all collation elements (work buffers) were represented in a sort key (target buffers).</p> <p>Action: Call the service again with new parameter value n in the target buffer.</p> | Collation |

Table 92. Return and reason codes from z/OS Unicode Services (continued)

| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
|----------------------------|----------------------------|---|------------|
| 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: <ul style="list-style-type: none"> • 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 data set. Action: Verify that the profile parameter on the call to the conversion services exists on the data set 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 92. Return and reason codes from z/OS Unicode Services (continued) | | | |
|--|-------------------------|--|---------------|
| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
| 8 | 17 | <p>Name: CUN_RS_INVALID_NORM_DATA_VER</p> <p>Meaning: Invalid Normalization data Version was introduced when trying to use the Normalization services.</p> <p>Action: Call the service again with a valid normalization data version (31/64-bit respectively):</p> <ul style="list-style-type: none"> • CUNBNPRM_NONE/CUN4BNPR_NONE • CUNBNPRM_UNI301/CUN4BNPR_UNI301 • CUNBNPRM_UNI320/CUN4BNPR_UNI320 • CUNBNPRM_UNI401/CUN4BNPR_UNI401 • CUNBNPRM_UNI410/CUN4BNPR_UNI410 • CUNBNPRM_UNI600/CUN4BNPR_UNI600 • CUNBNPRM_UNI900/CUN4BNPR_UNI900 • CUNBNPRM_UNIX13/CUN4BNPR_UNIX13 | Normalization |
| 8 | 18 | <p>Name: CUN_RS_INVALID_COLLATION_KEYWORD_VALUES</p> <p>Meaning: Invalid collation keyword values were introduced in CUN4BOPR_Collation_Keyword or CUNBOPRM_Collation_Keyword (31/64-bit respectively) collation parameter area field.</p> <p>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) in the collation parameter description topic.</p> | Collation |
| 8 | 19 | <p>Name: CUN_RS_INVALID_UCA_VERSION</p> <p>Meaning: Invalid Unicode collation version (or incompatible UCA version to the collation version) on fields: CUN4BOPR_UCA_Ver or CUNBOPRM_UCA_Ver (31/64-bit respectively)</p> <p>Action: Call the service again with a valid or compatible UCA version to the collation version (31/64-bit respectively):</p> <ul style="list-style-type: none"> • CUNBOPRM_UCAempty/CUN4BOPR_UCAempty • CUNBOPRM_UCA301/CUN4BOPR_UCA301 • CUNBOPRM_UCA400R1/CUN4BOPR_UCA400R1 • CUNBOPRM_UCA410/CUN4BOPR_UCA410 • CUNBOPRM_UCA600/CUN4BOPR_UCA600 • CUNBOPRM_UCA900/CUN4BOPR_UCA900 • CUNBOPRM_UCAX13/CUN4BOPR_UCAX13 | Collation |
| 8 | 1A | <p>Name: CUN_RS_INVALID_CASEFIST_VALUE</p> <p>Meaning: Invalid case first value.</p> <p>Action: Call the service again with a valid case first value:</p> <ul style="list-style-type: none"> • CASEFIRST_OFF • CASEFIRST_UPPER • CASEFIRST_LOWER | Collation |
| 8 | 1B | <p>Name: CUN_RS_INVALID_LOCALE_INPUT</p> <p>Meaning: Invalid locale input.</p> <p>Action: See Appendix E, “Locales for collation and case support,” on page 537 for valid locales support.</p> | Collation |
| 8 | 1C | <p>Name: CUN_RS_TARG_BUFF_LEN_ZERO</p> <p>Meaning: Target buffer length is 0.</p> <p>Action: Call the service again with new parameter value in the target buffer length.</p> | Stringprep |

| Table 92. Return and reason codes from z/OS Unicode Services (continued) | | | |
|--|-------------------------|--|------------|
| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
| 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 |
| 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_UNI_VERSION Meaning: An unsupported Unicode version was specified for CASE conversion service. Action: Specify one of the following: <ul style="list-style-type: none"> • CUNBAPRM_UNI300 / CUN4BAPR_UNI300 • CUNBAPRM_UNI301 / CUN4BAPR_UNI301 • CUNBAPRM_UNI320 / CUN4BAPR_UNI320 • CUNBAPRM_UNI401 / CUN4BAPR_UNI401 • CUNBAPRM_UNI410 / CUN4BAPR_UNI410 • CUNBAPRM_UNI500 / CUN4BAPR_UNI500 • CUNBAPRM_UNI600 / CUN4BAPR_UNI600 • CUNBAPRM_UNI900 / CUN4BAPR_UNI900 • CUNBAPRM_UNIX13 / CUN4BAPR_UNIX13 | Conversion |
| 8 | 21 | Name: CUN_RS_BIDI_CANNOT_SHAPE Meaning: Transformation stopped due to an input code element that cannot be shaped. Action: Call the service again with different input. | Conversion |
| 8 | 22 | Name: CUN_RS_BIDI_INCOMPLETE_COMPOSITE Meaning: Transformation stopped due to an incomplete composite sequence at the end of the source buffer. Action: Call the service again with different input. | Conversion |
| 8 | 23 | Name: CUN_RS_BIDI_RANGE_ERROR Meaning: More than 15 embedding levels are present, or the source buffer contains unbalanced directional layout information (push/pop), or an incomplete composite sequence has been detected in the beginning of the source buffer. Action: Call the service again with different input. | Conversion |
| 8 | 24 | Name: CUN_RS_BIDI_PARM_CONFLICT Meaning: The parameter values are set to a meaningless combination. Action: Call the service again with different input. | Conversion |

| Table 92. Return and reason codes from z/OS Unicode Services (continued) | | | |
|--|-------------------------|---|------------------------|
| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
| 8 | 25 | Name: CUN_RS_INVALID_BIDI_KEYWORD_VALUES Meaning: Invalid keyword values were introduced. Action: Call the service again with a valid keyword value. | Conversion |
| 8 | 26 | Name: CUN_RS_LOC_NOT_SUPPORTED Meaning: The locale name specified in the locale parameter is not supported. Action: Call the service again with a valid locale name. | Dynamic locale service |
| 8 | 27 | Name: CUN_RS_LOC_CCSID_NOT_SUPPORTED Meaning: The CCSID specified for the Targ_CCSID parameter is not supported. Action: Call the service again with a valid CCSID. | Dynamic locale service |
| 8 | 28 | Name: CUN_RS_LOC_BUILD_ERROR Meaning: An error was encountered while building the target locale. Action: Call the service again with different input. | Dynamic locale service |
| 8 | 29 | Name: CUN_RS_LOC_ENV_ERROR Meaning: An I/O error was encountered while building the target locale. Action: Check your file system environment and call the service again. | Dynamic locale service |
| 8 | 2A | Name: CUN_RS_LOC_DATA_FMT_NOT_SUPPORTED Meaning: The dynamic locale service does not support more than two byte codes when the CUNBLPRM_Flags1 Data_fmt bit is set to 1. Action: Call the service again with a valid CCSID. | Dynamic locale service |
| 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: <ul style="list-style-type: none"> • 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 |
| 0C | 4 | Name: CUN_RS_NO_MEM Meaning: Unable to allocate memory. Action: IPL is necessary to recover. | Conversion |
| 10 | 1 | Name: CUN_RS_INCONSISTENT_UCCB Meaning: The UCCB is in an inconsistent state. Action: IPL is necessary to recover. | Infrastructure |

| Table 92. Return and reason codes from z/OS Unicode Services (continued) | | | |
|--|-------------------------|--|----------------|
| Hexadecimal Return Code | Hexadecimal Reason Code | Name of reason code Meaning and Action | Component |
| 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. |
| 4 | Warnings issued | A duplicate statement has been ignored. Check the listing for the following messages: <ul style="list-style-type: none"> • CUN1027W • CUN1029W |

| Return Code | Meaning | Action |
|-------------|-----------------------------|---|
| 8 | User error | <p>The input (JCL or control statements) is incorrect. Check the listing for the following messages:</p> <ul style="list-style-type: none"> • CUN1003E • CUN1018E • CUN1019E • CUN1020E • CUN1021E • CUN1022E • CUN1023E • CUN1024E • CUN1025E • CUN1004E • CUN1006E • CUN1007E • CUN1008E • CUN1009E • CUN1010E • CUN1011E • CUN1012E • CUN1026E |
| 0C | Environment error | <p>An error occurred during the handling of a file or the work storage. Check the listing for the following messages:</p> <ul style="list-style-type: none"> • CUN1013E |
| 20 | Error recovery has occurred | <p>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.</p> |

Appendix I. Accessibility

Accessible publications for this product are offered through [IBM Documentation for z/OS \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS documentation see [How to Send Feedback to IBM](#) to leave documentation feedback.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

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
Site Counsel
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 document 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, 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.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS \(www.ibm.com/software/support/systemsz/lifecycle\)](http://www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [Copyright and Trademark information \(www.ibm.com/legal/copytrade.shtml\)](http://www.ibm.com/legal/copytrade.shtml).

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.

Unicode is a registered trademark of Unicode, Inc., in the United States and other countries.

Glossary of terms and abbreviations

This glossary defines technical terms and abbreviations used in Unicode documentation.

This glossary defines technical terms and abbreviations used in *z/OS Unicode Services User's Guide and Reference*.

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.

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

bidi

Bidirectional and character shaping service.

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.

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 6, “Collation,” on page 111](#) for more information.

collation rules

Rules which set the properties for Unicode strings. See [Chapter 6, “Collation,” on page 111](#) for more information.

composite conversion

Converting an 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.

LTR

Left-to-right character orientation.

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 5, "Normalization," on page 97](#).

normalization form

One of the four Unicode normalization forms defined in the Unicode Standard Annex #15, "Unicode Normalization Forms". See [Chapter 5, "Normalization," on page 97](#) 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 5, "Normalization," on page 97](#) for more information.

normalization form D (NFD)

The normalization form that results from the canonical decomposition of a Unicode string. See [Chapter 5, "Normalization," on page 97](#) 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 5, "Normalization," on page 97](#) for more information.

normalization form KD (NFKD)

The normalization form that results from the compatibility decomposition of a Unicode string. See [Chapter 5, "Normalization," on page 97](#) 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 data-stream

Note: If the conversion is to Unicode and the source code point is undefined by the source standard, round trip cannot be used. Because of this, the code point becomes a substitution code and does not round trip.

RTL

Right-to-left character orientation.

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” on page 163](#) 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.

UTF-EBCDIC

EBCDIC-friendly Unicode or **UCS transformation format**: 8-bit encoding form.

W

Weight

A value that identifies each part of the collation level for each Unicode character.

Index

Numerics

0 – 9: User-defined conversions, value for technique-character in CONVERSION [264](#)

A

accessibility
 contact IBM [611](#)

ACRI
 definition in glossary [617](#)
additional coding-related information
 see 'ACRI' definition in glossary [617](#)

address space, primary
 restriction while calling the case conversion services [81](#)
 restriction while calling the character conversion services [19](#)
 restriction while calling the conversion information service [192](#)

ALET
 specified for character conversion [17](#)

American National Standard Code for Information Interchange
 see 'ASCII' definition in glossary [617](#)

American National Standards Institute
 see 'ANSI' definition in glossary [617](#)

amode
 restriction while calling the case conversion services [81](#)
 restriction while calling the character conversion services [19](#)
 restriction while calling the conversion information service [192](#)

ANSI
 definition in glossary [617](#)

AR mode
 restriction while calling the case conversion services [81](#)
 restriction while calling the character conversion services [19](#)
 restriction while calling the conversion information service [192](#)

ASC mode
 restriction while calling the case conversion services [81](#)
 restriction while calling the character conversion services [19](#)
 restriction while calling the conversion information service [192](#)

ASCII
 definition in glossary [617](#)

assistive technologies [611](#)

authorization
 restriction while calling the case conversion services [81](#)
 restriction while calling the character conversion services [19](#)
 restriction while calling the conversion information service [192](#)

B

batch client example [281](#)
batch client, user interface [281](#)
batch tools, Unicode [281](#)
batch, send mail [283](#), [287](#)
bidi
 definition in glossary [617](#)
 indirect conversion [16](#)
Bidi
 general description [7](#)
bidirectional and character shaping transformation services [15](#)
big endian
 definition in glossary [617](#)

C

C interface
 mapping of parameters for case conversion [81](#)
 mapping of parameters for character conversion [19](#)
 mapping of parameters for conversion information service [192](#)
 mapping of parameters for normalization [99](#)

C interface for case conversion [81](#)

C interface for character conversion [19](#)

C interface for conversion information service [192](#)

C interface for normalization [98](#)

C: Customized Subset, value for technique-character in CONVERSION [264](#)

call syntax for case conversion [81](#), [83](#)

call syntax for character conversion [19](#), [24](#)

call syntax for conversion information service [192](#), [198](#)

call syntax for normalization [98](#), [100](#)

calling the stub routine
 for character conversion [6](#)

case conversion

 ALET [81](#)

 C interface, call syntax [81](#)

 C interface, mapping of parameters [81](#)

 CASE control statement [80](#)

 definition in glossary [617](#)

 general description [7](#)

 HLASM interface, call syntax [83](#)

 HLASM interface, mapping of parameters [84](#), [89](#)

 interfaces, description of parameters [85](#), [91](#)

 mapping of parameters, C interface [81](#)

 mapping of parameters, HLASM interface [84](#), [89](#)

 parameters in area CUN4BAPR [91](#)

 parameters in area CUNBAPRM [85](#)

 restrictions of the calling environment [81](#)

 return and reason codes [597](#)

 return codes, classification [597](#)

 stub routine [79](#)

case conversion handle

 CUN4BAPR_Conv_Type [92](#)

 CUNBAPRM_Conv_Type [86](#)

- CCSID
 - converting strings of text characters [15](#)
 - definition in glossary [617](#)
 - intermediate CCSID, definition in glossary [620](#)
 - range from [15](#)
- CCSIDs below [15](#)
- CDRA
 - definition in glossary [617](#)
- character
 - definition in glossary [617](#)
- character conversion
 - ALET [17](#)
 - C interface, call syntax [19](#)
 - C interface, mapping of parameters [19](#)
 - calling the stub routine [6](#)
 - conversion handle [17](#)
 - definition in glossary [618](#)
 - general description [15](#)
 - HLASM interface, call syntax [24](#)
 - HLASM interface, mapping of parameters [25](#), [35](#), [46](#), [61](#)
 - indirect conversion [16](#)
 - interfaces, description of parameters [28](#), [38](#), [48](#), [64](#)
 - items to be provided by caller [15](#)
 - mapping of parameters, C interface [19](#)
 - mapping of parameters, HLASM interface [25](#), [35](#), [46](#), [61](#)
 - parameters in area CUN4BCPR [38](#)
 - parameters in area CUN4BDPR [64](#)
 - parameters in area CUNBCPRM [28](#)
 - parameters in area CUNBDPRM [48](#)
 - restrictions of the calling environment [19](#)
 - return and reason codes [597](#)
 - return codes, classification [597](#)
 - UCCE, CUN4BCPR_Conv_Handle [39](#)
 - UCCE, CUNBCPRM_Conv_Handle [29](#)
- character conversion handle
 - case conversion [17](#)
- character data representation architecture
 - see 'CCSID' definition in glossary [617](#)
 - see 'CDRA' definition in glossary [617](#)
- character set
 - definition in glossary [618](#)
- code
 - definition in glossary [618](#)
- code page
 - definition in glossary [618](#)
 - source code page, see FROM-CCSID [17](#)
 - target code page, see TO-CCSID [17](#)
- code page conversion
 - definition in glossary [618](#)
- code page global identifier
 - see 'CPGID' definition in glossary [619](#)
- code point
 - definition in glossary [618](#)
- code set
 - definition in glossary [618](#)
- coded character
 - definition in glossary [618](#)
- coded character set
 - definition in glossary [618](#)
- codes
 - list of z/OS support for Unicode codes [597](#)
- collation
 - general description [7](#)
- collation (*continued*)
 - HLASM interface, mapping of parameters [127](#)
 - interfaces, description of parameters [131](#), [149](#)
 - mapping of parameters, HLASM interface [127](#)
 - parameters in area CUN4BOPR [149](#)
 - parameters in area CUNBOPRM [131](#)
- collation level
 - definition in glossary [618](#)
- collation rules
 - definition in glossary [618](#)
- comments in PARMDD [304](#)
- composite conversion
 - definition in glossary [618](#)
- composition [97](#)
- contact
 - z/OS [611](#)
- control character
 - definition in glossary [618](#)
- control function
 - definition in glossary [618](#)
- control parameters
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)
- conversion environment
 - definition in glossary [618](#)
- conversion handle
 - case conversion [17](#), [80](#)
 - case conversion, CUN4BAPR_Conv_Type [92](#)
 - case conversion, CUNBAPRM_Conv_Type [86](#)
 - critical case when invalid [267](#)
- conversion image
 - amount of storage needed [265](#)
 - creating [254](#)
 - definition in glossary [618](#)
 - general description [254](#)
 - loading into storage [254](#)
- Conversion information
 - general description [8](#)
- conversion information service
 - C interface, call syntax [192](#)
 - C interface, mapping of parameters [192](#)
 - CCSID information [191](#)
 - HLASM interface, call syntax [198](#)
 - HLASM interface, mapping of parameters [202](#), [216](#)
 - interfaces, description of parameters [207](#), [221](#)
 - mapping of parameters, C interface [192](#)
 - mapping of parameters, HLASM interface [202](#), [216](#)
 - parameters in area CUN4BIPR [221](#)
 - parameters in area CUNBIPRM [207](#)
 - restrictions of the calling environment [192](#)
 - stub routine [191](#)
- conversion of data between specified CCSIDs [15](#)
- conversion service [191](#)
- conversion services
 - definition in glossary [619](#)
- conversion tables
 - input to image generator [254](#)
 - provided by Unicode Consortium [7](#)
- conversion to upper or lower case [79](#)
- conversion type
 - in character conversion, CUN4BAPR_Conv_Type [92](#)

conversion type (*continued*)
 in character conversion, CUNBAPRM_Conv_Type [86](#)
 CPGID
 definition in glossary [619](#)
 cross memory mode
 restriction while calling the case conversion services [81](#)
 restriction while calling the character conversion services [19](#)
 restriction while calling the conversion information service [192](#)
 CUN_RC_ENV_ERR, return code [597](#)
 CUN_RC_OK, return code [597](#)
 CUN_RC_SYS_ERR, return code [597](#)
 CUN_RC_USER_WARN, return code [597](#)
 CUN_RC_WARN, return code [597](#)
 CUN4BAPR parameter area for case conversion [91](#)
 CUN4BCPR parameter area for character conversion [38](#)
 CUN4BDPR parameter area for character conversion [64](#)
 CUN4BIPR parameter area for conversion information service [221](#)
 CUN4BIPR_CCSID1_ES [221](#), [223](#)
 CUN4BNPR parameter area for normalization [106](#)
 CUN4BOPR parameter area for collation [149](#)
 CUNBA_DDA_req, constant [80](#)
 CUNBAPRM parameter area for case conversion [85](#)
 CUNBAPRM_Src_Buf_ALET [86](#)
 CUNBAPRM_Src_Buf_Len [86](#)
 CUNBAPRM_Targ_Buf_ALET [86](#)
 CUNBAPRM_Targ_Buf_Len [86](#)
 CUNBAPRM_Targ_Buf_Ptr [86](#)
 CUNBCPRM parameter area for character conversion [28](#)
 CUNBCPRM_Designator [33](#)
 CUNBDPRM parameter area for character conversion [48](#)
 CUNBIPRM parameter area for conversion information service [207](#)
 CUNBIPRM_CCSID1_ES [208](#), [210](#)
 CUNBN_DDA_req, constant [98](#)
 CUNBNPRM parameter area for normalization [102](#)
 CUNBOPRM parameter area for collation [131](#)
 CUNMCBLI [289](#)
 CUNMCBLI example [297](#)
 CUNMCBLI user interface [297](#)
 CUNMCSMH [286](#)
 CUNMCSMM [298](#)
 CUNMCSMX [283](#)
 CUNMCSMX user interface [284](#)
 CUNMRCSH [286](#)
 CUNMRCSH example [287](#)
 CUNMRCSH return codes [289](#)
 CUNMRCSH user interface [287](#)
 CUNMRCSH, JCL example [287](#)
 CUNMRCSH, send mail from batch [287](#)
 CUNMRCSM [298](#)
 CUNMRCSM enhancements [299](#)
 CUNMRCSM examples [306](#)
 CUNMRCSM JCL example [299](#)
 CUNMRCSM return codes [306](#)
 CUNMRCSM user interface [300](#)
 CUNMRCSM, example [299](#)
 CUNMRCSM, send mail from batch [298](#)
 CUNMRCSX [283](#)
 CUNMRCSX example [284](#)
 CUNMRCSX return codes [286](#)
 CUNMTUNI [281](#)

D

DBCS
 definition in glossary [619](#)
 indirect conversion [16](#)
 DDA, see dynamic data area
 constant CUNBAPRM_DDA_Req [80](#)
 constant CUNBNPRM_DDA_Req [98](#)
 decomposition [97](#)
 designator sequence
 definition in glossary [619](#)
 direct conversion
 definition in glossary [619](#)
 dispatchable unit mode
 restriction while calling the case conversion services [81](#)
 restriction while calling the character conversion services [19](#)
 restriction while calling the conversion information service [192](#)
 double-byte (coded) character set
 see 'DBCS' definition in glossary [619](#)
 dynamic data area
 case conversion [80](#), [98](#)
 Dynamic locale service
 general description [8](#)

E

E: Enforced Subset, value for technique-character in CONVERSION [264](#)
 EBCDIC
 definition in glossary [619](#)
 empty conversion environment
 definition in glossary [619](#)
 empty image
 definition in glossary [619](#)
 encoding scheme
 definition in glossary [619](#)
 encoding scheme identifier
 definition in glossary [619](#)
 endian
 definition in glossary [619](#)
 enforced subset
 definition in glossary [619](#)
 EUC
 definition in glossary [619](#)
 example JCL, CUNMCBLI [297](#)
 example JCL, CUNMRCSX [284](#)
 example, batch client [281](#)
 example, CUNMRCSH [287](#)
 example, CUNMRCSM [299](#)
 examples, CUNMRCSM [306](#)
 external interfaces for case conversion [81](#), [83](#)
 external interfaces for character conversion [19](#), [24](#)
 external interfaces for conversion information service [192](#), [198](#)
 external interfaces for normalization [98](#), [100](#)

F

FROM-CCSID
 definition [17](#)
 definition in glossary [619](#)

FROM-CCSID (*continued*)
in character conversion [17](#)

G

GB18030
definition in glossary [620](#)
graphic character
definition in glossary [620](#)
graphic character set
definition in glossary [620](#)

H

HASN mode
restriction while calling the case conversion services [81](#)
restriction while calling the character conversion services [19](#)
restriction while calling the conversion information service [192](#)
high-surrogate
definition in glossary [620](#)
HLASM interface
mapping of parameters for case conversion [84](#), [89](#)
mapping of parameters for character conversion [25](#), [35](#), [46](#), [61](#)
mapping of parameters for collation [127](#)
mapping of parameters for conversion information service [202](#), [216](#)
mapping of parameters for normalization [101](#), [105](#)
HLASM interface for case conversion [83](#)
HLASM interface for character conversion [24](#)
HLASM interface for conversion information service [198](#)
HLASM interface for normalization [100](#)

I

IBM Extended Binary Coded Decimal Interchange Code
see 'EBCDIC' definition in glossary [619](#)
IDF
see interface definition file [620](#)
IEASYSxx
editing [253](#)
image generator
definition in glossary [620](#)
input [254](#)
return and reason codes [608](#)
indirect conversion
between mixed code pages and anything else than SBCS [16](#)
between TBCS and anything else than DBCS [16](#)
between UTF-8 and anything else than DBCS [16](#)
definition in glossary [620](#)
infrastructure
definition in glossary [620](#)
interface definition file
constant CUNBN_DDA_req [98](#)
CUNBNIDF [98](#)
definition in glossary [620](#)
intermediate CCSID
definition in glossary [620](#)
interrupt status
restriction while calling the case conversion services [81](#)

interrupt status (*continued*)
restriction while calling the character conversion services [19](#)
restriction while calling the conversion information service [192](#)
invalid conversion handle [267](#)

J

JCL example, CUNMRCSH [287](#)
JCL example, CUNMRCSM [299](#)

K

keyboard
navigation [611](#)
PF keys [611](#)
shortcut keys [611](#)
knowledge base
input to image generator [254](#)

L

L: Language Environment-Behavior, value for technique-character in CONVERSION [264](#)
little endian
definition in glossary [620](#)
locks
restriction while calling the character conversion services [19](#)
Locks
restriction while calling the case conversion services [81](#)
restriction while calling the conversion information service [192](#)
low-surrogate
definition in glossary [620](#)
lowercase
definition in glossary [620](#)
LTR
definition in glossary [620](#)

M

M: Modified for special use, value for technique-character in CONVERSION [264](#)
malformed character
definition in glossary [621](#)
mapping of parameters
for case conversion [81](#), [84](#), [89](#)
for character conversion [19](#), [25](#), [35](#), [46](#), [61](#)
for collation [127](#)
for conversion information service [192](#), [202](#), [216](#)
for normalization [99](#), [101](#), [105](#)
mapping of parameters in C interface, for case conversion [81](#)
mapping of parameters in C interface, for character conversion [19](#)
mapping of parameters in C interface, for conversion information service [192](#)
mapping of parameters in C interface, for normalization [99](#)
mapping of parameters in HLASM interface, for case conversion [84](#), [89](#)
mapping of parameters in HLASM interface, for character conversion [25](#), [35](#), [46](#), [61](#)

- mapping of parameters in HLASM interface, for collation [127](#)
- mapping of parameters in HLASM interface, for conversion information service [202](#), [216](#)
- mapping of parameters in HLASM interface, for normalization [101](#), [105](#)
- MBCS
 - definition in glossary [620](#)
 - internal handling of MBCS conversions
 - detailed description [333](#)
 - illustration of MBCS decomposition [332](#)
- mixed code page
 - definition in glossary [620](#)
- mixed code pages [309](#)
- mode
 - amode [19](#), [81](#), [192](#)
 - AR mode [19](#), [81](#), [192](#)
 - ASC mode [19](#), [81](#), [192](#)
 - cross memory mode [19](#), [81](#), [192](#)
 - dispatchable unit mode [19](#), [81](#), [192](#)
 - HASN mode [19](#), [81](#), [192](#)
 - PASN mode [19](#), [81](#), [192](#)
 - SASN mode [19](#), [81](#), [192](#)
- multi-byte character set
 - see 'MBCS' definition in glossary [620](#)

N

- navigation
 - keyboard [611](#)
- normalization
 - C interface, call syntax [98](#)
 - C interface, mapping of parameters [99](#)
 - definition in glossary [621](#)
 - general description [7](#)
 - HLASM interface, call syntax [100](#)
 - HLASM interface, mapping of parameters [101](#), [105](#)
 - interfaces, description of parameters [102](#), [106](#)
 - mapping of parameters, C interface [99](#)
 - mapping of parameters, HLASM interface [101](#), [105](#)
 - parameters in area CUN4BNPR [106](#)
 - parameters in area CUNBNPRM [102](#)
 - stub routine [97](#)
- normalization form
 - definition in glossary [621](#)
- normalization form C
 - definition in glossary [621](#)
- normalization form D
 - definition in glossary [621](#)
- normalization form KC
 - definition in glossary [621](#)
- normalization form KD
 - definition in glossary [621](#)

P

- parameter
 - description of parameters, case conversion [85](#), [91](#)
 - description of parameters, character conversion [28](#), [38](#), [48](#), [64](#)
 - description of parameters, collation [131](#), [149](#)
 - description of parameters, conversion information service [207](#), [221](#)
 - description of parameters, normalization [102](#), [106](#)

- parameter area
 - CUN4BAPR for case conversion [91](#)
 - CUN4BCPR for character conversion [38](#)
 - CUN4BDPR for character conversion [64](#)
 - CUN4BIPR for conversion information service [221](#)
 - CUN4BNPR for normalization [106](#)
 - CUN4BOPR for collation [149](#)
 - CUNBAPRM for case conversion [85](#)
 - CUNBCPRM for character conversion [28](#)
 - CUNBDPRM for character conversion [48](#)
 - CUNBIPRM for conversion information service [207](#)
 - CUNBNPRM for normalization [102](#)
 - CUNBOPRM for collation [131](#)
- PARMDD comments [304](#)
- PASN mode
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)
- PC
 - definition in glossary [621](#)
- personal computer
 - see 'PC' definition in glossary [621](#)
- primary address space
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)
- problem determination
 - invalid conversion handle [267](#)
 - target buffer overflow [76](#)
 - work buffer overflow [98](#)
- problem state
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the conversion information service [192](#)
- programing interfaces for case conversion [81](#), [83](#)
- programing interfaces for character conversion [19](#), [24](#)
- programing interfaces for conversion information service [192](#), [198](#)
- programing interfaces for normalization [98](#), [100](#)
- PSW key
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)

Q

- QBCS
 - definition in glossary [621](#)
- quadruple-byte character set
 - see 'QBCS' definition in glossary [621](#)

R

- R: Roundtrip, value for technique-character in CONVERSION [264](#)
- range of CCSIDs
 - range from [15](#)

- reason code
 - reason code CUN_RS_TRG_EXH (target buffer exhausted) [76](#)
- reason codes
 - conversion services [597](#)
- recommendations
 - conversion handle [267](#)
 - for the calling environment (case conversion) [81](#)
 - for the calling environment (character conversion) [19](#)
 - for the calling environment (conversion information service) [192](#)
 - target buffer size [76](#)
 - work buffer size [98](#)
- recovery environment
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)
- restrictions
 - case conversion [81](#)
 - character conversion [19](#)
 - conversion information service [192](#)
- return codes
 - conversion services [597](#)
 - image generator [608](#)
- return codes, CUNMRCSH [289](#)
- return codes, CUNMRCSM [306](#)
- return codes, CUNMRCSX [286](#)
- Round trip
 - definition in glossary [621](#)
- RTL
 - definition in glossary [621](#)

S

- sample program
 - case conversion [95](#)
 - character conversion [78](#)
 - collation [166](#)
 - conversion information service [229](#)
 - normalization [108](#)
 - stringprep conversion [189](#)
- SASN mode
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)
- SBCS
 - definition in glossary [621](#)
 - indirect conversion [16](#)
- script
 - definition in glossary [622](#)
- send mail from batch [283](#), [287](#)
- send mail from batch, CUNMRCSM [298](#)
- sending HTML data [286](#)
- sending MIME data [283](#), [298](#)
- sending MIME data using COBOL API [289](#)
- shortcut keys [611](#)
- simple code page
 - definition in glossary [622](#)
- simple code pages [309](#)
- simple conversion

- simple conversion (*continued*)
 - definition in glossary [622](#)
- single-byte character set
 - see 'SBCS' definition in glossary [621](#)
- SMTP, COBOL API, sending MIME data [289](#)
- SMTP, sending HTML data [286](#)
- SMTP, sending MIME data [283](#), [298](#)
- sort key
 - definition in glossary [622](#)
- source code page, see FROM-CCSID [17](#)
- SRB or task
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)
- storage
 - needed for a conversion image [265](#)
- Stringprep
 - general description [7](#)
- stub routine
 - CUN4LCNV [15](#)
 - CUNLASE [79](#)
 - CUNLCNV [15](#)
 - CUNLINFO [191](#)
 - CUNLNORM [97](#)
- sub code page
 - definition in glossary [622](#)
- summary of changes [xix](#)
- supervisor state
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the conversion information service [192](#)
- surrogate pair [622](#)

T

- target buffer
 - calculating the size [76](#)
 - reason code CUN_RS_TRG_EXH (target buffer exhausted) [76](#)
- target code page, see TO-CCSID [17](#)
- task or SRB
 - restriction while calling the case conversion services [81](#)
 - restriction while calling the character conversion services [19](#)
 - restriction while calling the conversion information service [192](#)
- TBCS
 - definition in glossary [622](#)
 - indirect conversion [16](#)
- technique
 - definition in glossary [622](#)
- TO-CCSID
 - definition [17](#)
 - definition in glossary [622](#)
 - in character conversion [17](#)
- triple-byte character set
 - see 'TBCS' definition in glossary [622](#)
- two-stage conversion, see indirect conversion [16](#)

U

UCAE
 definition in glossary [622](#)

UCCB
 definition in glossary [622](#)

UCCE
 character conversion, CUN4BCPR_Conv_Handle [39](#)
 character conversion, CUNBCPRM_Conv_Handle [29](#)
 definition in glossary [622](#)

UCS
 definition in glossary [622](#)

UCS transformation format
 see 'UTF-16' definition in glossary [623](#)
 see 'UTF-32' definition in glossary [623](#)
 see 'UTF-8' definition in glossary [623](#)
 see 'UTF-EBCDIC' definition in glossary [623](#)

UCS-2
 definition in glossary [622](#)

Unicode
 ASCII [4](#)
 EBCDIC [4](#)
 environment [11](#), [250](#)
 sample code [14](#)
 standard [3](#)

Unicode batch tools [281](#)

Unicode character conversion control entry
 see 'UCCE' definition in glossary [622](#)

Unicode Consortium
 conversion tables provided by [7](#)

UNICODE SERVICES batch client [281](#)

Unicode standard
 range of CCSIDs from [15](#)

Unicode Standard
 definition in glossary [622](#)

Unicode transformation format
 see 'UTF-16' definition in glossary [623](#)
 see 'UTF-32' definition in glossary [623](#)
 see 'UTF-8' definition in glossary [623](#)
 see 'UTF-EBCDIC' definition in glossary [623](#)

universal character set
 see 'UCS' definition in glossary [622](#)

uppercase
 definition in glossary [622](#)

user interface
 ISPF [611](#)
 TSO/E [611](#)

user interface, batch client [281](#)

user interface, CUNMCBLI [297](#)

user interface, CUNMCSMX [284](#)

user interface, CUNMRCSH [287](#)

user interface, CUNMRCSM [300](#)

user-defined CCSID
 valid range [15](#)

UTF-16
 definition in glossary [623](#)

UTF-32
 definition in glossary [623](#)

UTF-8
 definition in glossary [623](#)
 indirect conversion [16](#)

UTF-EBCDIC
 definition in glossary [623](#)

W

weight
 definition in glossary [623](#)

work buffer
 calculating the size [98](#)

Z

z/OS support for Unicode
 case conversion, general description [7](#)
 conversion to upper or lower case [7](#)
 MBCS conversions, general description [329](#)
 prerequisites [253](#)
 return and reason codes [597](#)
 return and reason codes from conversion services [597](#)
 return codes from image generator [608](#)



Product Number: 5655-ZOS

SA38-0680-70

