z/OS V1R13

BCP Unicode: BiDi phase two support

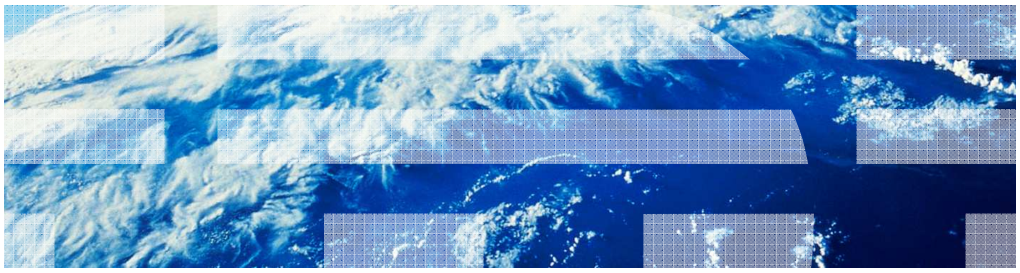


Table of contents

Session objectives

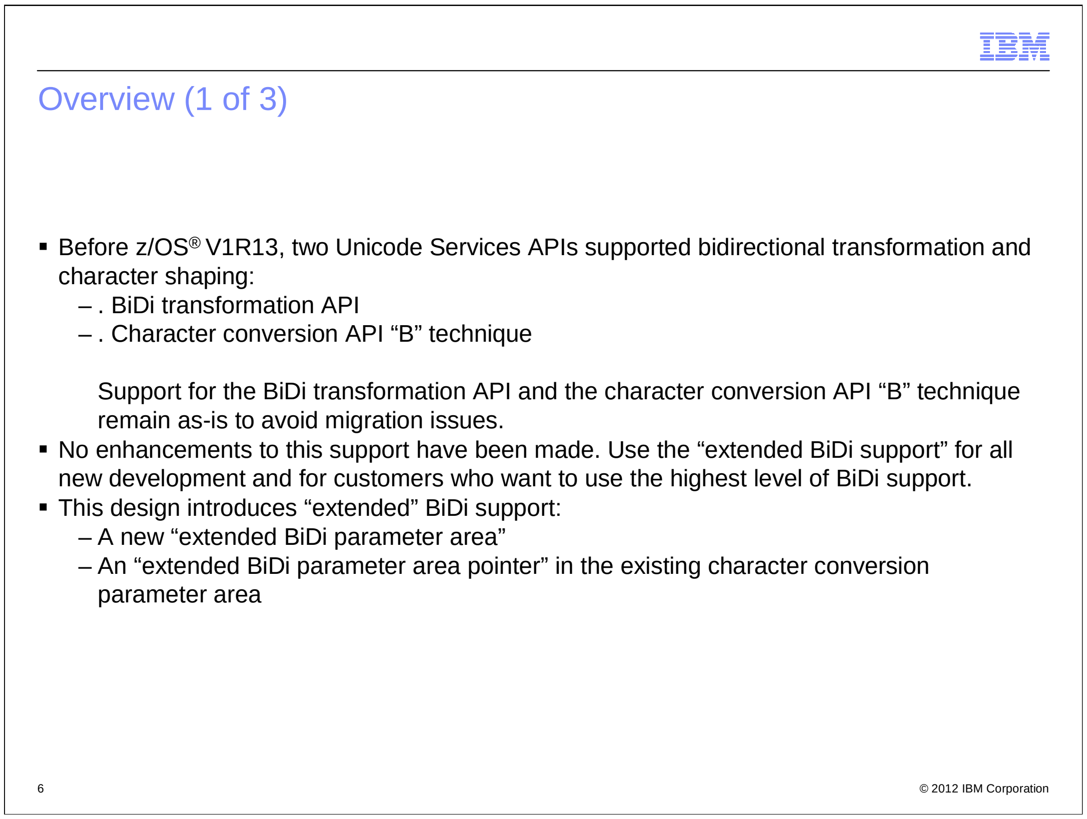
- The objective of this presentation is to describe the enhancements that were made to the Unicode services character conversion service. These enhancements provide support for bidirectional transformation and character shaping (BiDi).

Overview (1 of 2)

- Problem statement / Need addressed
  - The Unicode services character conversion service's support for bidirectional transformation and character shaping (BiDi) is outdated.
- Solution
  - The Unicode services character conversion service API has been enhanced with the highest level of BiDi support available. The new BiDi support meets some of the standards set forth in the Unicode consortium’s standard annex #9. The annex can be found at the Unicode consortium’s website at:
    http://www.unicode.org/reports/tr9
    The extended BiDi support does not implement the full Unicode consortium’s BiDi standard. It implements the highest level of support available.

Overview (2 of 2)

- Benefit / value
  - Users of Unicode services character conversion service's bidirectional transformation and character shaping (BiDi) support receive the highest level of support available



Overview (1 of 3)

- Before z/OS® V1R13, two Unicode Services APIs supported bidirectional transformation and character shaping:
  - . BiDi transformation API
  - . Character conversion API “B” technique

    Support for the BiDi transformation API and the character conversion API “B” technique remain as-is to avoid migration issues.
- No enhancements to this support have been made. Use the “extended BiDi support” for all new development and for customers who want to use the highest level of BiDi support.
- This design introduces “extended” BiDi support:
  - A new “extended BiDi parameter area”
  - An “extended BiDi parameter area pointer” in the existing character conversion parameter area

Overview (2 of 3)

- CUNSISM7 in dataset SYS1.SAMPLIB is a sample that demonstrates how to use the Unicode services extended BiDi support
- CUNSISM8 in dataset SYS1.SAMPLIB is another sample that demonstrates how to use the Unicode services extended BiDi support. This sample uses the open group's standard “portable layout services” interface functions:

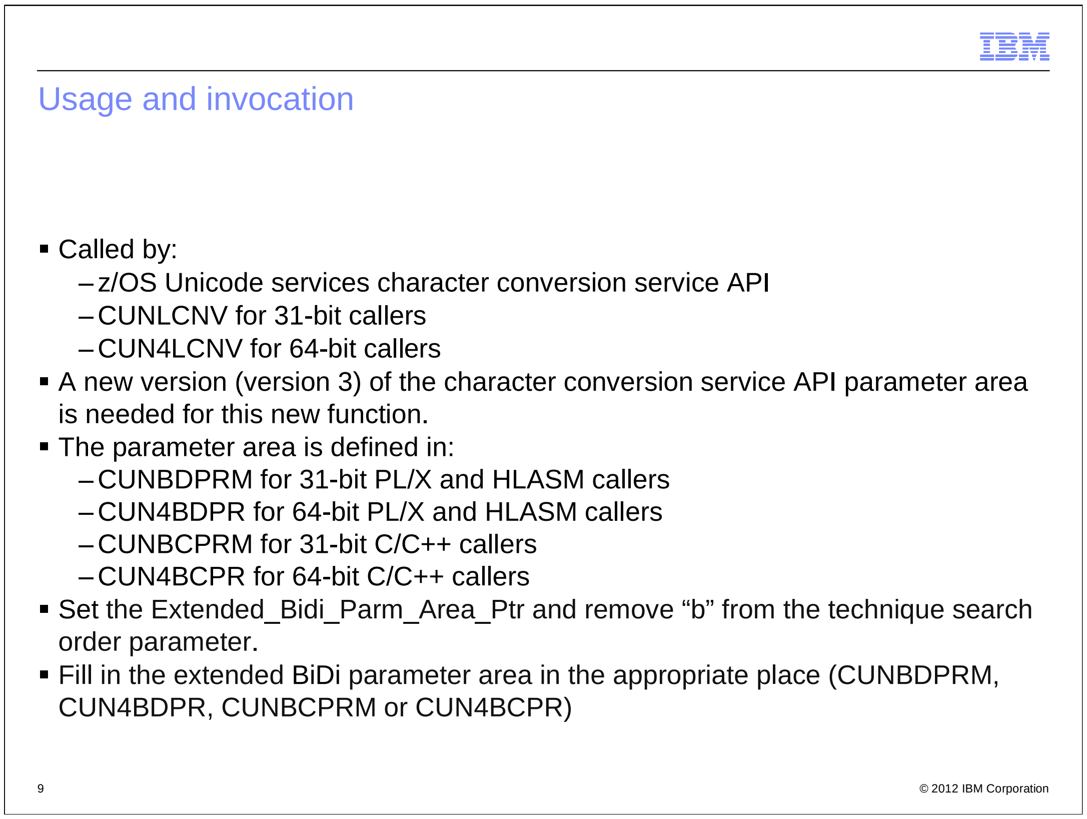
  m_create_layout() - Create and initialize a layout object
  m_getvalues_layout() - Query layout values of a layout object
  m_setvalues_layout() - Set layout values of a layout object
  m_getoptions_layout() - Query the current setting of layout options of a layout object
  m_setoptions_layout() - Change the layout options of a layout object
  m_transform_layout() - Layout transformation for character strings
  m_wtransform_layout() - Layout transformation for Wide-Character strings
  m_getprocessedlength_layout() - Query the length of source text processed by the last transform operation on a layout object
  m_destroy_layout() - Destroy a layout object

  Note: This support is intended to run in an LE environment as a replacement to LE BiDi support.

Overview (3 of 3)

- Additional notes:
  - The Unicode services character conversion service does not support the use of user-customized character conversion tables while using the extended BiDi support.
  - The Unicode services extended BiDi support operates on CCSID 01200. If the source and target CCSIDs are not both 01200 (or equivalent CCSIDs), the BiDi algorithm will perform a two-stage conversion, regardless of any other considerations. The source buffer is first converted to CCSID 01200, BiDi transformations are performed, then the characters are converted to the target CCSID. The work buffer (Wrk_Buf) buffer is required for this.
  - The Unicode services extended BiDi support can only be used for some CCSIDs. The supported CCSIDs are Arabic and Hebrew CCSIDs 00420, 00424, 00425, 00856, 00862, 00864, 00916, 01046, 01089, 01255 or 01256. Using it for an unsupported CCSID will result in one of the following errors:

    RC = CUN_RC_USER_ERR
    RS = CUN_RS_CCSID_NOT_SUPP



Usage and invocation

- Called by:
  - z/OS Unicode services character conversion service API
  - CUNLCNV for 31-bit callers
  - CUN4LCNV for 64-bit callers
- A new version (version 3) of the character conversion service API parameter area is needed for this new function.
- The parameter area is defined in:
  - CUNBDPRM for 31-bit PL/X and HLASM callers
  - CUN4BDPR for 64-bit PL/X and HLASM callers
  - CUNBCPRM for 31-bit C/C++ callers
  - CUN4BCPR for 64-bit C/C++ callers
- Set the Extended_Bidi_Parm_Area_Ptr and remove “b” from the technique search order parameter.
- Fill in the extended BiDi parameter area in the appropriate place (CUNBDPRM, CUN4BDPR, CUNBCPRM or CUN4BCPR)

Usage and invocation ... (1 of 61)

- Updates to parameter area for C/C++ - CUNBCPRM structure

```
typedef struct tagCUNBCPRM {
  long Version; /* Structure version number */

  ...

  long         Return_Code;
  long         Reason_Code;
  unsigned int Res6;
  struct {
     int       ETF3E_Behavior        :  1,
                                     : 15;
  } Flag3;
  char         Res7[2];
  CUNBDPRM *   Extended_Bidi_Parm_Area_Ptr;
  char         Res8[64];
  } CUNBCPRM;
```

Usage and invocation ... (2 of 61)

- Parameter area for C/C++ - CUNBCPRM structure

```
/* 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;
```

continued on next page

Usage and invocation ... (3 of 61)

- Parameter area for C/C++ - CUNBDPRM structure (continued)

```
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             ShapeCharsetSize_Front;
 int             ShapeCharsetSize_Back;
 int             CheckMode;
 unsigned int    InpBufIndex;
 unsigned long   Streaming_Processed_Length;
 int             ArabicOneCellShaping_Src;
 int             ArabicOneCellShaping_Targ;
 int             WordBreak_Src;
 int             WordBreak_Targ;
```

Continued on next page

Usage and invocation ... (4 of 61)

- Parameter area for C/C++ - CUNBDPRM structure (continued)

```
  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];

} CUNBDPRM;
```

Usage and invocation ... (5 of 61)

- Updates to parameter area for C/C++ - CUN4BCPR structure

```
typedef struct tagCUN4BCPR {
  unsigned int  Version; /* Structure version number */

  ...

  unsigned int  Return_Code;
  unsigned int  Reason_Code;
  int           Res4;
  long          Res5;
  struct {
     int       ETF3E_Behavior        :  1,
                                     : 15;
  } Flag3;
  char         Res7;
  CUN4BDPR *   Extended_Bidi_Parm_Area_Ptr;
  char         Res8[64];
  } CUN4BCPR;
```

Usage and invocation ... (6 of 61)

- Parameter area for C/C++ - CUN4BDPR structure

```
/* the extended BiDi parameter area */

 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;
```

Continued on next page

Usage and invocation ... (7 of 61)

- Parameter area for C/C++ - CUN4BDPR structure (continued)

```
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             ShapeCharsetSize_Front;
 int             ShapeCharsetSize_Back;
 int             CheckMode;
 unsigned long   InpBufIndex;
 unsigned long   Streaming_Processed_Length;
 int             ArabicOneCellShaping_Src;
 int             ArabicOneCellShaping_Targ;
 int             WordBreak_Src;
 int             WordBreak_Targ;
```

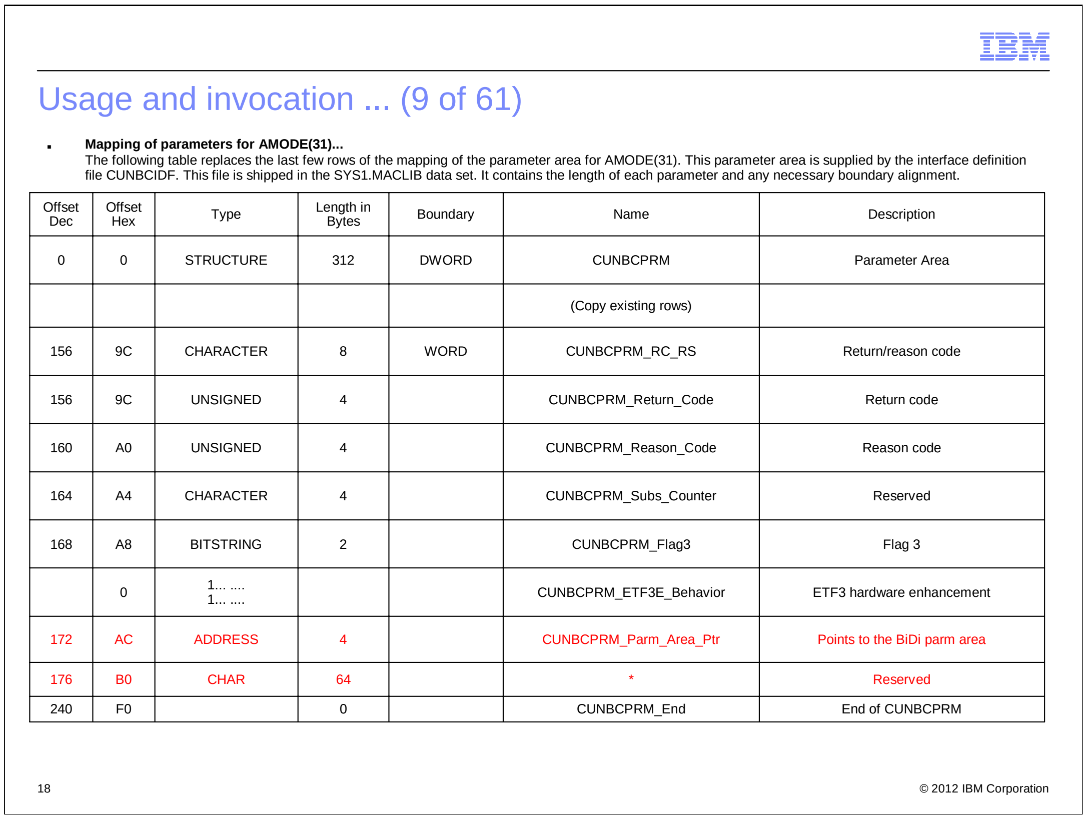
Continued on next page

Usage and invocation ... (8 of 61)

- Parameter area for C/C++ - CUN4BDPR structure (continued)

```
  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;
```



IBM

Usage and invocation ... (9 of 61)

- **Mapping of parameters for AMODE(31)...**
  The following table replaces the last few rows of the mapping of the parameter area for AMODE(31). This parameter area is supplied by the interface definition file CUNBCIDF. This file is shipped in the SYS1.MACLIB data set. It contains the length of each parameter and any necessary boundary alignment.

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Description
0	0	STRUCTURE	312	DWORD	CUNBCPRM	Parameter Area
					(Copy existing rows)	
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
	0	1... .... 1...			CUNBCPRM_ETF3E_Behavior	ETF3 hardware enhancement
172	AC	ADDRESS	4		CUNBCPRM_Parm_Area_Ptr	Points to the BiDi parm area
176	B0	CHAR	64		*	Reserved
240	F0		0		CUNBCPRM_End	End of CUNBCPRM

18

Usage and invocation ... (10 of 61)

- **Mapping of Parameters for AMODE(64)...**

  The following table replaces the last few rows of the existing mapping of the parameter area for AMODE(64). This parameter area is supplied by the interface definition file CUN4BCID. This file is shipped in the SYS1.MACLIB data set. It contains the length of each parameter and any necessary boundary alignment.

Offset Dec	Offset Hex	Type	Length in Bytes	Boundary	Name	Description
0	0	STRUCTURE	342	DWORD	CUN4BCPR	Parameter Area
					(Copy existing rows)	
180	B4	UNSIGNED	4		CUN4BCPR_Return_Code	Return code
184	B8	UNSIGNED	4		CUN4BCPR_Reason_Code	Reason code
188	BC	CHARACTER	8		CUN4BCPR_Subs_Counter	Reserved
196	C4	BITSTRING	2		CUN4BCPR_Flag3	Flag 3
	0	1...			CUN4BCPR_ETF3E_Behavior	ETF3 hardware enhancement
198	C6	CHAR	6		*	Reserved
204	CC	ADDRESS	8		CUN4BCPR_Parm_Area_Ptr	Points to the BiDi parm area
212	D4	CHAR	64		*	Reserved
276	114		0		CUN4BCPR_End	End of CUN4BCPR

Usage and invocation ... (11 of 61)

- **Description of Parameters in area CUNBCPRM**
  Only new and changed items are presented here.

  ========> Changes to existing fields <========

  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. Use 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_Technique - set by caller
Specifies the technique-search-order for the given CCSID pair. See Understanding how Unicode Services loads conversion tables. In addition to the techniques search orders (R,E,C,L,M and 0-9), now 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-CSSID 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, 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.

Usage and invocation ... (12 of 61)

- **Description of Parameters in area CUNBCPRM (Continued)**
  Only new and changed items are presented here.

  ========> Changes to existing fields <========

  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_Bidi_Context – set by caller

  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)

Usage and invocation ... (13 of 61)

- **Description of Parameters in area CUNBCPRM (Continued)**
  Only new and changed items are presented here.

  ========> Changes to existing fields <========

  CUNBCPRM_Bidi_ImpAlg – set by caller

  Specifies the algorithm to be used if technique B was specified. This field is for the B technique.

  0: Indicates that the basic algorithm will be used

  1: Indicates that the implicit algorithm will be used

  ========> New fields <========

  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.

Usage and invocation ... (14 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)...**
  The HLASM mapping of the extended BiDi parameter area is given in interface definition files CUNBCIDF for 31-bit in dataset SYS1.MACLIB.

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
0	STRUCTURE		DWORD	CUNBDPRM	Extended BiDi parameter area
0	UNSIGNED	4		CUNBDPRM_Version	Version of the parameter area
4	UNSIGNED	4		CUNBDPRM_Length	Length, in bytes, of the parameter area
8	BITSTRING	1		CUNBDPRM_InFlags	Input flags
	1...			CUNBDPRM_XOpen_Defaults	Specifies X/Open portable layout option defaults
	.1..			CUNBDPRM_KBS_Defaults	Specifies Unicode Services knowledge base defaults
	..1.			CUNBDPRM_Keyword	Specifies BiDi keyword
	...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...			CUNBDPRM_Layout_Roundtrip	Specifies if round trip processing is to be used
	.1..			CUNBDPRM_Layout_WinCompat	Specifies if WinCompat mode is to be used

Usage and invocation ... (15 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
	..1.			CUNBDPRM_Layout_ImpToImp	Specifies if a “Logical to Logical” transformation is to be performed
	...1			CUNBDPRM_Layout_Remove_Marks	Specifies if all BiDi marks will be removed
	.... 1...			CUNBDPRM_Layout_Insert_Marks	Specifies if BiDi marks are to be inserted
	.... .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	CHAR	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

Usage and invocation ... (16 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
1C	UNSIGNED	4		CUNBDPRM_Context_Targ	Context of the target buffer
20	UNSIGNED	4		CUNBDPRM_TypeOfText_Src	Type of text of the source buffer
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

IBM

Usage and invocation ... (17 of 61)

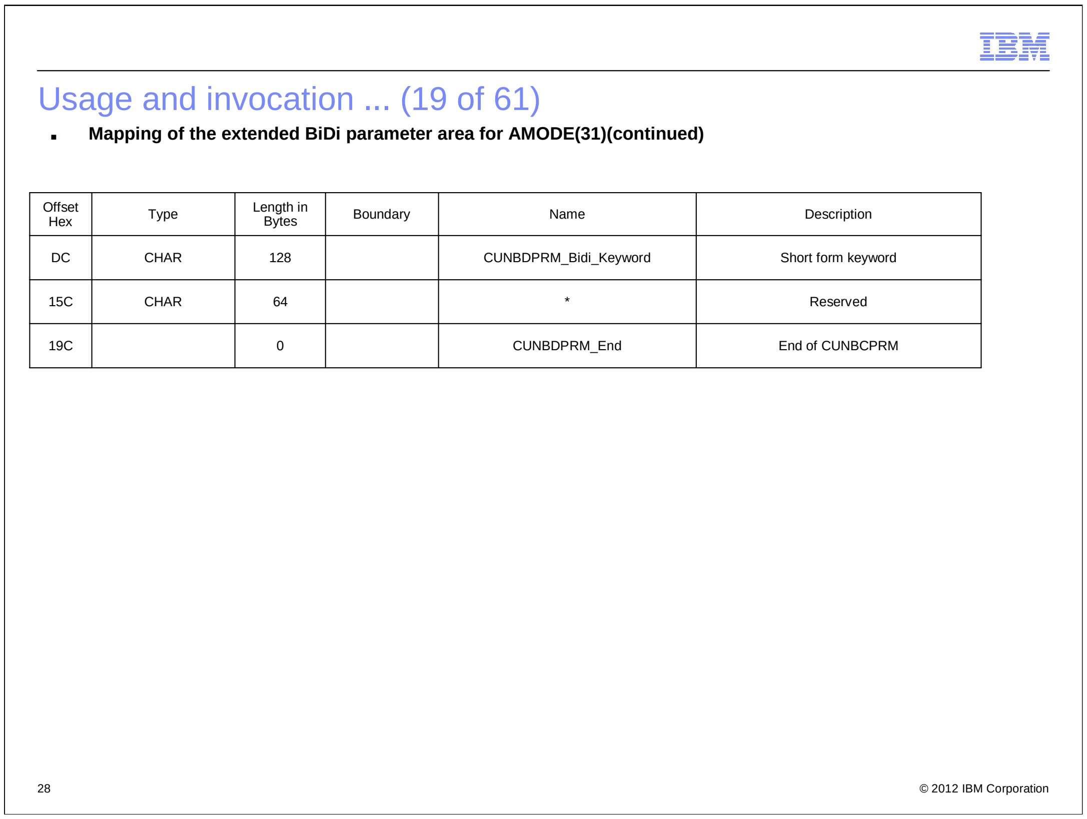
- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
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	Lam-Alef edit mode used in the source buffer

Usage and invocation ... (18 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
74	UNSIGNED	4		CUNBDPRM_LamAlefEditMode_Targ	Lam-Alef 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
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	BiDi pointer
9C	CHAR	64		CUNBDPRM_Layout_Streaming_State	State of the layout streaming operation



Usage and invocation ... (19 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
DC	CHAR	128		CUNBDPRM_Bidi_Keyword	Short form keyword
15C	CHAR	64		*	Reserved
19C		0		CUNBDPRM_End	End of CUNBCPRM

Usage and invocation ... (20 of 61)

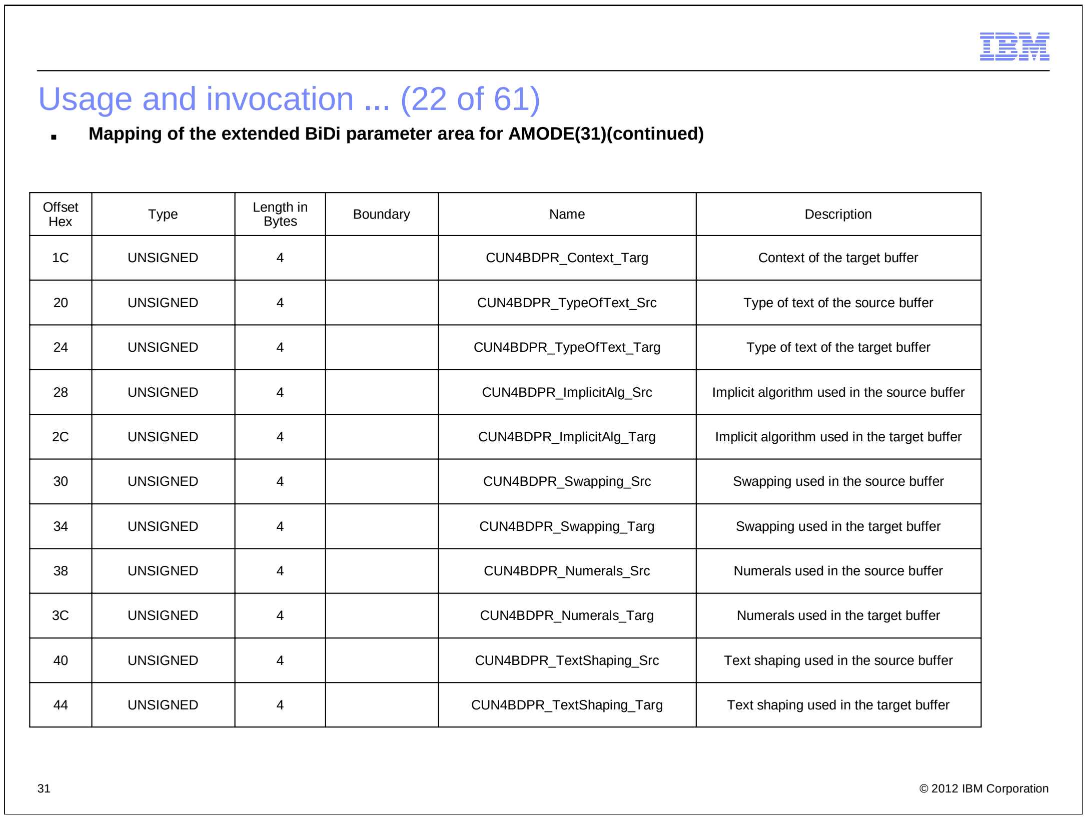
- **Mapping of the extended BiDi parameter area for AMODE(64)...**
  The HLASM mapping of the extended BiDi parameter area is given in interface definition file CUN4BCID for 64-bit in dataset SYS1.MACLIB.

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
0	STRUCTURE		DWORD	CUN4BDPR	Extended BiDi parameter area
0	UNSIGNED	4		CUN4BDPR_Version	Version of the parameter area
4	UNSIGNED	4		CUN4BDPR_Length	Length, in bytes, of the parameter area
8	BITSTRING	1		CUN4BDPR_InFlags	Input flags
	`1... ....`			CUN4BDPR_XOpen_Defaults	Specifies X/Open portable layout option defaults
	`.1.. ....`			CUN4BDPR_KBS_Defaults	Specifies Unicode Services knowledge base defaults
	`..1. ....`			CUN4BDPR_Keyword	Specifies BiDi keyword
	`...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... ....`			CUN4BDPR_Layout_Roundtrip	Specifies if round trip processing is to be used
	`.1.. ....`			CUN4BDPR_Layout_WinCompat	Specifies if WinCompat mode is to be used

Usage and invocation ... (21 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
	`..1. ....`			CUN4BDPR_Layout_ImpToImp	Specifies if a “Logical to Logical” transformation is to be performed
	`...1 ....`			CUN4BDPR_Layout_Remove_Marks	Specifies if all BiDi marks will be removed
	`.... 1...`			CUN4BDPR_Layout_Insert_Marks	Specifies if BiDi marks are to be inserted
	`.... .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	CHAR	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

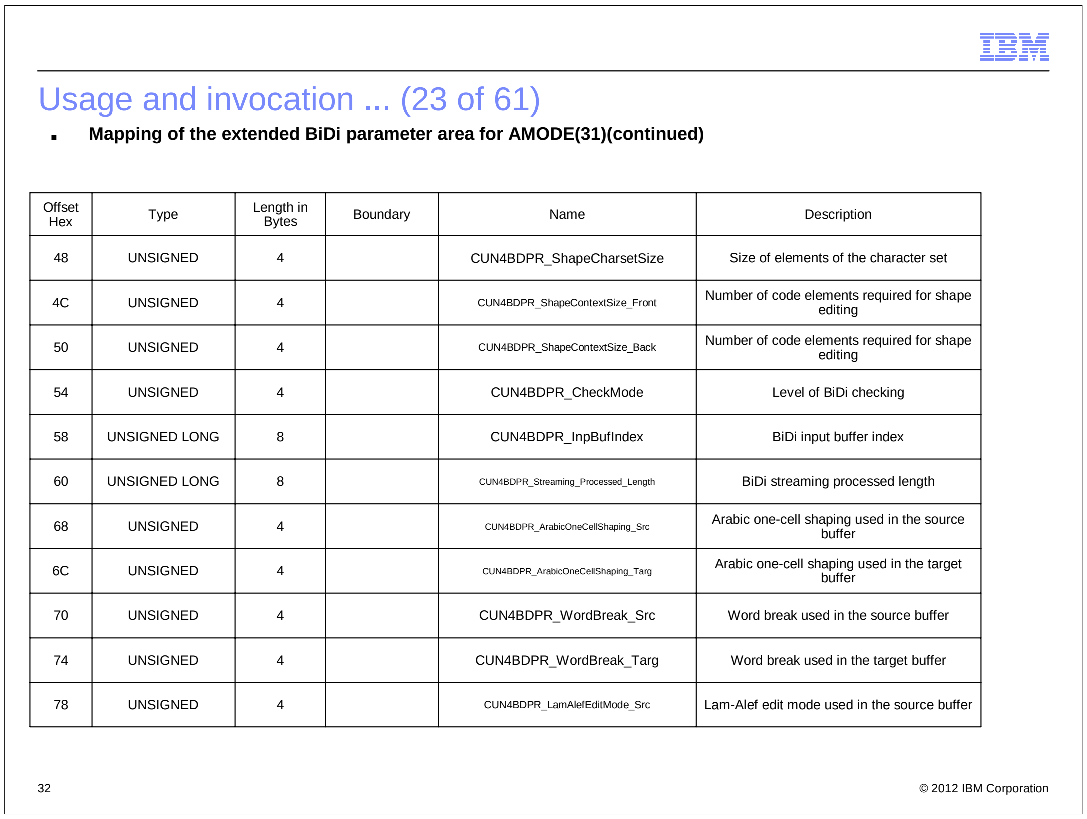


Usage and invocation ... (22 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
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
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

31 © 2012 IBM Corporation

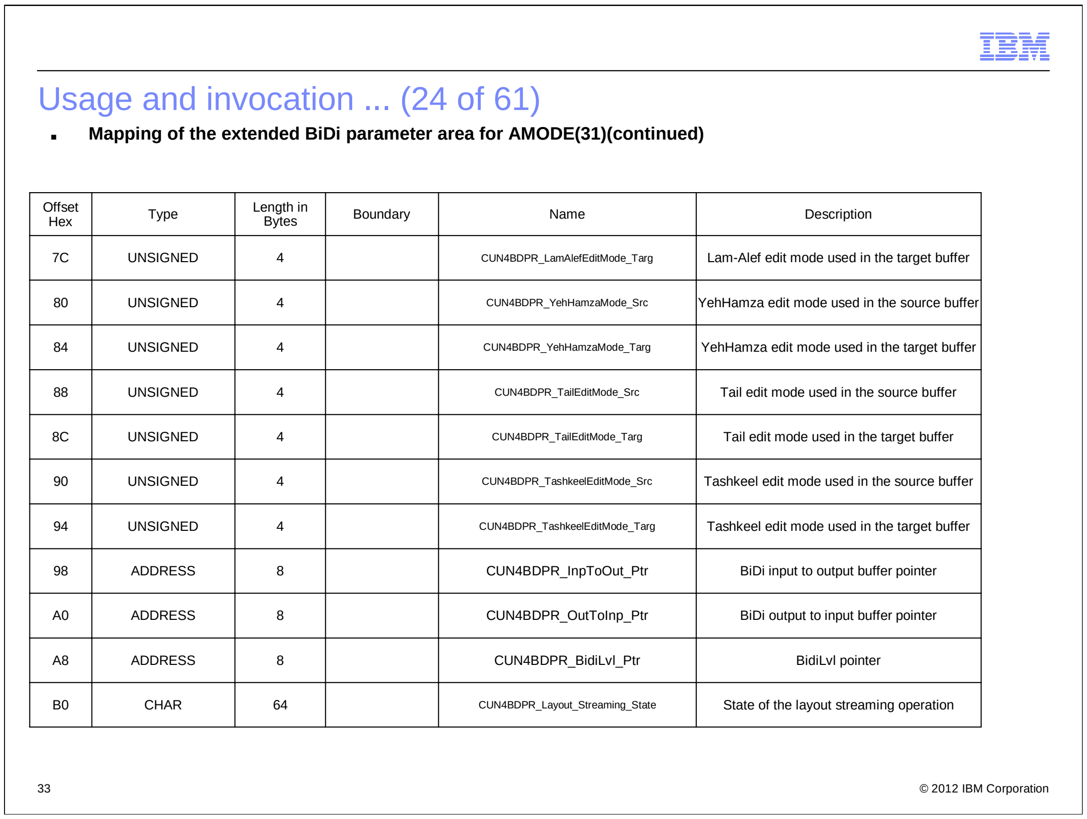


Usage and invocation ... (23 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
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 LONG	8		CUN4BDPR_InpBufIndex	BiDi input buffer index
60	UNSIGNED LONG	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	Lam-Alef edit mode used in the source buffer

32 © 2012 IBM Corporation



Usage and invocation ... (24 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
7C	UNSIGNED	4		CUN4BDPR_LamAlefEditMode_Targ	Lam-Alef 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
A8	ADDRESS	8		CUN4BDPR_BidiLvl_Ptr	BidiLvl pointer
B0	CHAR	64		CUN4BDPR_Layout_Streaming_State	State of the layout streaming operation

33

Usage and invocation ... (25 of 61)

- **Mapping of the extended BiDi parameter area for AMODE(31)(continued)**

Offset Hex	Type	Length in Bytes	Boundary	Name	Description
F0	CHAR	128		CUN4BDPR_Bidi_Keyword	Short form keyword
170	CHAR	64		*	Reserved
1B0		0		CUN4BDPR_End	End of CUNBCPRM

Usage and invocation ... (26 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR**

  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

Usage and invocation ... (27 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_XOpen_Defaults - set by caller

  Specifies whether 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 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.

Usage and invocation ... (28 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_Keyword - set by caller

  Specifies whether 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

Usage and invocation ... (29 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_Layout_Roundtrip - set by caller

  Specifies whether numbers located between LTR text and RTL text are associated with the RTL text. This makes the algorithm reversible. Reversible algorithms enable the round trip (from visual to logical and back to visual) 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 whether the algorithm used to perform BiDi transformations should approximate the algorithm used in Microsoft® Windows® XP, rather than strictly conforming to the Unicode BiDi algorithm. Possible values are:
  0: Do not approximate the Microsoft algorithm
  1: Approximate the Microsoft algorithm

Usage and invocation ... (30 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_Layout_ImpToImp - set by caller

  Specifies whether to perform a logical to logical transformation:

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

  This mode may be needed when Arabic or Hebrew logical text (possibly including numbers or phrases in English) has to be displayed in 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 English logical text (possibly including phrases in Arabic or Hebrew) has to be displayed in 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 whether to remove all BiDi marks (LRM or RLM) from the output text when performing a transformation. Possible values are:
  0: Do no remove BiDi marks from the output text.
  1: Remove BiDi marks 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. If both are set, this overrides Layout_Insert_Marks.

Usage and invocation ... (31 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_Layout_Insert_Marks - set by caller

  Specifies whether to insert BiDi marks (LRM or RLM) as 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. The minimum number of LRM or RLM characters will be added to the source text after reordering it so as to ensure the round trip. In other words, if the inverse transformation were performed on the resulting implicit text, with removal of BiDi marks (option Layout_Remove_Marks), the resulting text would be identical to the source text in the first transformation. The LRM and RLM characters that are added to 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.
  Ignored if specified together with CUNBDPRM_Layout_Remove_Marks.

  CUNBDPRM_Layout_Streaming - set by caller

  Specifies whether to use layout streaming. Layout streaming processes large text objects into parts using the piece by piece technique. The caller is responsible for concatenating the results of the successive calls. 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. To determine how much of the source text has been processed, read the value in CUNBDPRM_Streaming_Processed_Length immediately after performing the transform. Then resubmit any source text beyond that length in a subsequent transform operation. 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. You need to check for such an occurrence and perform one of the following actions:

  - 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.

Usage and invocation ... (32 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

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 whether 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 whether 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 before rendering the text.
0: The BiDi transformation does not require shape editing
1: The BiDi transformation requires shape editing

Usage and invocation ... (33 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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 which orientation is used when no strong character occurs 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.

Usage and invocation ... (34 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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 can be seen as if there were no sub segments.
  TEXT_IMPLICIT: Code elements are stored in logically ordered segments. Either, the characters are stored in the order they are pronounced when reading. Or, they are stored in the order in which characters are entered from a keyboard. The logical order (or logical sequence) of characters is needed for processing purposes, for example, 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. There are several possible algorithms for implicit reordering of directional segments. Therefore, the ImplicitAlg value is used to indicate the actual algorithm used (when TypeOfText is set to TEXT_IMPLICIT).
  TEXT_EXPLICIT: Code elements are stored in logically ordered segments with a set of embedded controls. The explicit algorithm eliminates the ambiguities 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.

Usage and invocation ... (35 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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: Use an implementation-defined implicit directional algorithm to reorder directional code elements when converting to or from an implicit form.
  The basic algorithm (used when ImplicitAlg is set to ALGOR_BASIC) is an implicit algorithm. However, it recognizes some control characters. Therefore it can be used when the TypeOfText descriptor is set to TEXT_EXPLICIT.

  Note: When TEXT_EXPLICIT is used in conjunction with ALGOR_BASIC, the controls can temporarily change the values of swapping, numerals and TextShaping. In general, do not set TypeOfText=TEXT_EXPLICIT and ImplicitAlg=ALGOR_IMPLICIT. The exception is if the ALGOR_IMPLICIT value equals ALGOR_BASIC for a given implementation.

  ALGOR_BASIC: Use the basic algorithm.

  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.

Usage and invocation ... (36 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_Numerals_Src - set by caller
  CUNBDPRM_Numerals_Targ - set by caller

  Specifies the shaping of numerals. Possible values are:
  NUMERALS_NOMINAL: Perform nominal shaping of numerals using the portable character set (Arabic numerals)
  NUMERALS_NATIONAL: Perform national shaping of numerals based on the script of the C locale
  NUMERALS_CONTEXTUAL: Perform contextual shaping of numerals depending on the context (script) of surrounding text. Examples are Hindi numbers in Arabic text and Arabic numbers otherwise.

  The default (for the C locale) is NUMERALS_NOMINAL.

Usage and invocation ... (37 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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

Usage and invocation ... (38 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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.

Usage and invocation ... (39 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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 stand-alone 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. It will follow 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 varies depending on locale-specific validation or assumptions.
  The reordering of bidirectional text will assign the nesting level of an unstabilized neutral character. It will follow 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.

Usage and invocation ... (40 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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 is converted to one character only which is the three quarters shape seen.

  Other settings: Seen tail characters remain as is.

Usage and invocation ... (41 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_WordBreak_Src - set by caller
  CUNBDPRM_WordBreak_Targ - set by caller

  Specifies whether to transform each word in isolation, independent of adjacent words, based on whitespace delimiters. Combinations are:
  WordBreak_Src is NO_BREAK, and WordBreak_Targ is BREAK: Transform each word in isolation, independent of adjacent words, based on whitespace delimiters.
  Other settings: Do not transform each word in isolation, independent of adjacent words, based on whitespace delimiters.

Usage and invocation ... (42 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_LamAlefEditMode_Src - set by caller
  CUNBDPRM_LamAlefEditMode_Targ - set by caller

  Specifies which Lam-Alef edit mode transformations to perform.
  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 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 end of the buffer.

  LamAlefEditMode_Src is LamAlefOff, and LamAlefEditMode_Targ is LamAlefAuto: For each Lam-Alef character found, expand it using space at end of the buffer. If there is no space at the end, use space at beginning of the buffer. If there is no space at the beginning, use space nearby (for example, the space after the Lam-Alef character).

  Other settings: Lam Alef characters remain as is.

Usage and invocation ... (43 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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

  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.

Usage and invocation ... (44 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_TashkeelEditMode_Src - set by caller
  CUNBDPRM_TashkeelEditMode_Targ - set by caller

  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 re-seize 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 re-seize 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 re-seize the data buffer.

  The default value for CUNBDPRM_TashkeelEditMode is TASHKEELEND.

Usage and invocation ... (45 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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 pointed to by InpBufIndex. It does not necessarily start from the beginning of Src_Buf.

  If not a NULL pointer, InpToOut_Ptr points to an array of values. The number of values in this array matches the number of bytes in Src_Buf, starting with the one pointed by InpBufIndex and going 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 multi-byte encoding, the index (for each byte of a code element in the Src_Buf) indicates the first byte of the transformed code element in the Targ_Buf.

  Specify NULL for InpToOut if you do not want an index array from Src_Buf to Targ_Buf.

  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 pointed to by InpBufIndex. It does not necessarily start from the beginning of Src_Buf.

  If not a NULL pointer, OutToInp_Ptr points to an array of values. The number of values in this array matches the 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 multi-byte encoding, the index (for each byte of a code element in the Targ_Buf) indicates the first byte of the source of the transformed code element in the Src_Buf.

  Specify NULL for OutToImp if you do not want an index array from Targ_Buf to Src_Buf.

Usage and invocation ... (46 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  CUNBDPRM_BidiLvl_Ptr - Set by caller

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

  If this argument is not a NULL pointer, it points to an array of values. This array has 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). They will indicate NestingLevels. Possible values are 0 to 15. This value 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 is ignored. The leftmost bit of each BidiLvl byte will contain a new cell indicator for composed character environments. It will have a value of either one (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 are required, set the argument property to NULL.

Usage and invocation ... (47 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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.

  You should set this area to all zero bytes the first time you call the service with Layout_Streaming. Then, do 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. Do 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.

Usage and invocation ... (48 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

  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...

  Notes:
  Most attributes (except for LayoutOptions and CheckMode attributes) can apply to both the source and target data. Therefore, the second letter in the *key* indicates whether the attribute is for the source (S) or target (T) buffer.

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

  In the example:

  OS0_OT1_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.

Usage and invocation ... (49 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
LayoutOptions	Lx	0-252	Layout options. Values: • 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 Example of Roundtrip and ImpToImp (or Logical to Logical): L160 For long path equivalent setting, see CUNBDPRM_Layout _Options description.

Usage and invocation ... (50 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
Orientation	Obx	0-4	The direction of the text. Values: • 0 = ORIENTATION_LTR (Input/Output Default) • 1 = ORIENTATION_RTL • 2 = ORIENTATION_TTBLR • 3 = ORIENTATION_TTBRL • 4 = ORIENTATION _CONTEXTUAL The mappings between short form and long form are defined by BIDI_ORIENTATION in the interface definition file CUNBCIDF.

Usage and invocation ... (51 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
Context	Cbx	0-1	Contextual orientation when the orientation attribute is set to ORIENTATION_CONTEXTUAL. Values: • 0 = CONTEXT_LTR (Input/Output Default) • 1 = CONTEXT_RTL The mappings between short form and long form are defined by BIDI_CONTEXT in the interface definition file CUNBCIDF.
TypeofText	Tbx	0-2	Type of the text. Values: • 0 = TEXT_VISUAL (Output default) • 1 = TEXT_IMPLICIT (Input default) • 2 = TEXT_EXPLICIT The mappings between short form and long form are defined by BIDI_TEXT_TYPE in the interface definition file CUNBCIDF.

Usage and invocation ... (52 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
ImplicitAlg	Ibx	0-1	Implicit algorithm used in the source/target buffer. Values: • 0 = ALGOR_BASIC (Input/Output Default) • 1 = ALGOR_IMPLICIT The mappings between short form and long form are defined by BIDI_IMPALG in the interface definition file CUNBCIDF.
Swapping	Sbx	0-1	Specifies whether symmetric swapping is enabled. Values: • 0 = SWAPPING_NO (Output default) • 1 = SWAPPING_YES (Input default) The mappings between short form and long form are defined by BIDI_SWAPPING in the interface definition file CUNBCIDF.

Usage and invocation ... (53 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
Numerals	Nbx	0-3	How numerals are shaped. Values: • 0 = NUMERALS_NOMINAL (Input default. Output default in Hebrew locale.) • 1 = NUMERALS_NATIONAL • 2 = NUMERALS_CONTEXTUAL (Output default in Arabic locale) • 3 = NUMERALS_NONE The mappings between short form and long form are defined by BIDI_NUMERALS in the interface definition file CUNBCIDF.

Usage and invocation ... (54 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
TextShaping	Ebx	0-7	Specifies whether text to be shaped. Values: • 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 The mappings between short form and long form are defined by BIDI_SHAPING in the interface definition file CUNBCIDF.

Usage and invocation ... (55 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
CheckMode	Hx	0-1	Level of BiDi checking (apply to both source and target). Values: • 0 = MODE_STREAM • 1 = MODE_EDIT (Input/Output default) The mappings between short form and long form are defined by BIDI_CHECKMODE in the interface definition file CUNBCIDF.
WordBreak	Wbx	0-1	Word break. Values: • 0 = WORD_BREAK • 1 = NO_BREAK (Input/Output default) The mappings between short form and long form are defined by BIDI_WORDBREAK in the interface definition file CUNBCIDF.

Usage and invocation ... (56 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
LamALefEdit	Fbx	0-5	Lab-Alef edit mode. Values: • 0 = LamAlefOn • 1 = LamAlefBegin • 2 = LamAlefResize • 3 = LamAlefNear • 4 = LamAlefAuto (Input/Output default) • 5 = LamAlefOff The mappings between short form and long form are defined by BIDI_LAMALEF in the interface definition file CUNBCIDF.
ArabicOneCell	Abx	0-1	Arabic one-cell shaping. Values: • 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.

Usage and invocation ... (57 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
TailMode	Mbx	0-1	Tail edit mode. Values: • 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: • 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.

Usage and invocation ... (58 of 61)

- **Description of parameters in area CUNBDPRM and CUN4BDPR (continued)**

Attribute name	**Format key: b=buffer (S=source or T=target) x=attribute value**	**Possible attribute values**	**Description**
YehHamza	Ybx	0-1	YehHamza edit mode. Values: • 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.

Usage and invocation ... (59 of 61)

- **New return and reason codes**

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Name of reason code** **Meaning and action**	**Component**
8	20	Name: CUN_RS_INVALID_UNI_VERSION Meaning:: Invalid Unicode Version was specified. Action: Call the service again with a valid Unicode Version	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

Usage and invocation ... (60 of 61)

- **New return and reason codes**

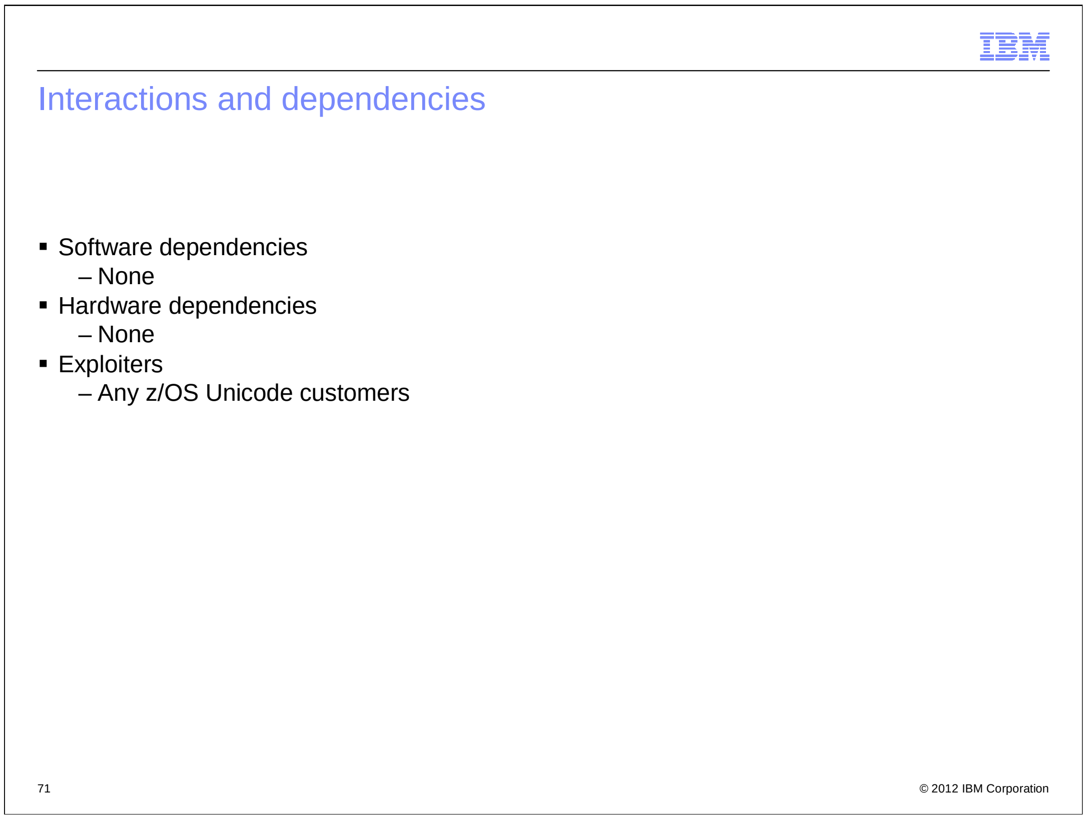
Hexadecimal Return Code	**Hexadecimal Reason Code**	**Name of reason code Meaning and action**	**Component**
8	24	Name: CUN_RS_BIDI_PARM_CONFLICT Meaning:: The parameter values are set to a meaningless combination. Action: Call the service again with different input.	Conversion
8	25	Name: CUN_RS_INVALID_BIDI_KEYWORD_VALUES Meaning: Invalid Keyword Values were introduced. Action: Call the service again with a valid keyword value	Conversion

Usage and invocation ... (61 of 61)

- **Mapping errno Values When Using Sample Object CUNSISM9 in Dataset SYS1.SAMPLIB. See Sample CUNSISM8 for Information Regarding How to Use Sample CUNSISM9**

  The Open Group's layout functions set errno values. These errno values are listed in the table below along with an indication of how they are handled by Unicode Services.

Errno value	**Unicode Services RC/RS**
EILSEQ	New RC/RS 8/21
E2BIG	Existing RC/RS 4/1
EINVAL	New RC/RS 8/22
ERANGE	New RC/RS 8/23
EBADF	New RC/RS 8/24



Interactions and dependencies

- Software dependencies
  - – None
- Hardware dependencies
  - – None
- Exploiters
  - – Any z/OS Unicode customers

Migration and coexistence considerations

- None

Installation

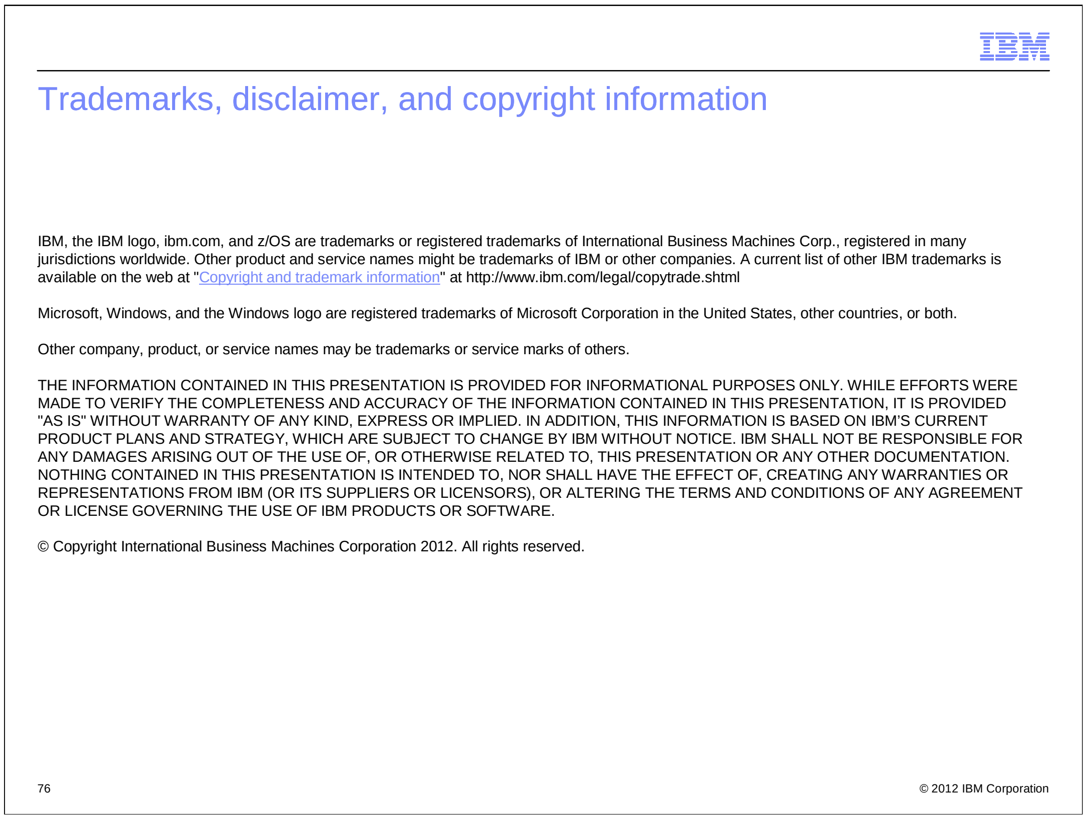
- None

Session summary

- With this implementation, z/OS Unicode services character conversion service has been enhanced with the highest level of BiDi support available.
- Before z/OSV1R13, two Unicode services APIs supported bidirectional transformation and character shaping:
  – BiDi transformation API
  – Character conversion API “b” technique
  – Support for the BiDi transformation API and the character conversion API “b” technique remain as-is to avoid migration issues.
- For specific BiDi behavior questions please contact Waleed Oransa in IBM Egypt at WORANSA@eg.ibm.com

Appendix - References

- Related materials for quick reference
  - z/OS Support for Unicode: Using Unicode Services (SA22-7649)



Trademarks, disclaimer, and copyright information