IBM i globalization

Supplement V2.1

June 2017

Table of Contents

About this information

This publication provides supplemental information for using IBM i globalization support.

You should see the primary information for i globalization support located in the IBM Knowledge Center http://www.ibm.com/support/knowledgecenter/ssw_ibm_i/welcome

IBM i globalization

https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/nls/rbagsprintingme.htm

Globalization reference information Supplements.

CCSID related updates

Changes to the CCSID values defined on IBM i

This table updates the table "**CCSID values defined on IBM i"** in the publication of the CCSIDs that are defined on the IBM i operating system.

CCSID	Encoding	Description
1371	1301	Traditional Chinese host mixed for additional CNS 11643 subset including 6395 UDC. (2016 updated version. CCSID 9563 level)

For information on other CCSIDs not listed visit the IBM CCSID web information:
http://www.ibm.com/software/globalization/ccsid/ccsid_registered.html
(This site lists all CCSIDs – even those not supported on IBM i.)

Other updates

LGSort API

There is an error in the description of the LGSORT API, found here:
https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/apis/QLGSORT.htm

Format of Request Control Block

The description for the "**Format of Request Control Block**" has an error in it.
The offset from "**Length of input file entry"** should be as follows:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of request control block
4	4	BINARY(4)	Type of request
8	8	BINARY(4)	Reserved
12	C	BINARY(4)	Options
16	10	BINARY(4)	Record length

20	14	BINARY(4)	Record count
24	18	BINARY(4)	Offset to key list
28	1C	BINARY(4)	Number of keys
32	20	BINARY(4)	Offset to national language sort information
36	24	BINARY(4)	Offset to input file list
40	28	BINARY(4)	Number of input files
44	2C	BINARY(4)	Offset to output file list
48	30	BINARY(4)	Number of output files
52	34	BINARY(4)	Length of key entry
56	38	BINARY(4)	Length of national language sort sequence information
60	**3C**	BINARY(4)	Length of input file entry
64	**40**	BINARY(4)	Length of output file entry
68	**44**	BINARY(4)	Offset to null byte map
72	**48**	BINARY(4)	Offset to variable length record access information
76	**4C**	BINARY(4)	Reserved

Length of request control block

The “Length of request control block” text is incorrect as to the minimum size needed.

It should read as follows:

Length of request control block.

*The total length of the request control block. The minimum size is **100** bytes, which allows for one key in the key list, no input or output files, and no national language sort sequence information. An error is returned if the length specified is less than the minimum.*

Bidirectional Language String Types

In the “Bidirectional Language String Types and Associated Attributes” table the entry for 10 is not correct.

It should have Arabic for numeric shaping as shown below:

String Type	Text Type	Numeric Shaping	Orientation	Text Shaping	Symmetrical Swapping
4	Visual	passthrough	LTR	Shaped	Off
5	Implicit	Arabic	LTR	Unshaped	On
6	Implicit	Arabic	RTL	Unshaped	On
8	Visual	passthrough	RTL	Shaped	Off
9	Visual	passthrough	RTL	Shaped	On
10	Implicit	Arabic	Contextual Left	Unshaped	On

There should also be a note at the end of the table that says:
The IBM Bidi competency center has determined that string type 10 should be used for bidirectional scripts encoded in Unicode where a string type may be needed.

PUA mapping support

An update to the text that describes this support to make it clear that the support is for mapping both directions.

Background
In Ideographic languages, such as Chinese, a need to create custom ideographs to represent new words not in the current standards exists. This custom ideograph is supported by “user-defined characters” that extend an encoding standard. EBCDIC calls these custom ideographs user-defined characters (UDC) and the standard has an allocated range for them. Unicode calls these custom ideographs private use area (PUA) and also has a range that is defined for them. These definitions do not use the same hexadecimal ranges. In addition, these user-defined characters are not standardized and their usage is unique by user. Therefore, user-defined characters can cause issues when mapping between EBCDIC and Unicode. To simplify this text, the use of the abbreviation of “PUA” is used to mean both EBCDIC UDC and Unicode PUA since the system support is the same except for the first line in the source text file.