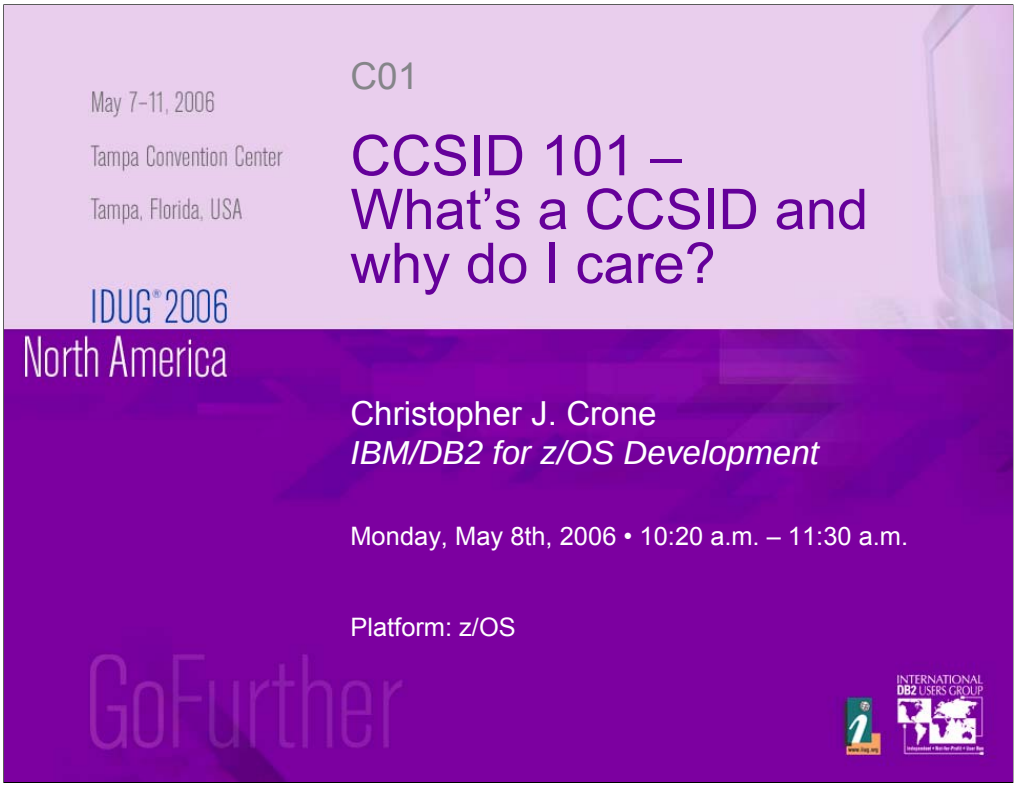
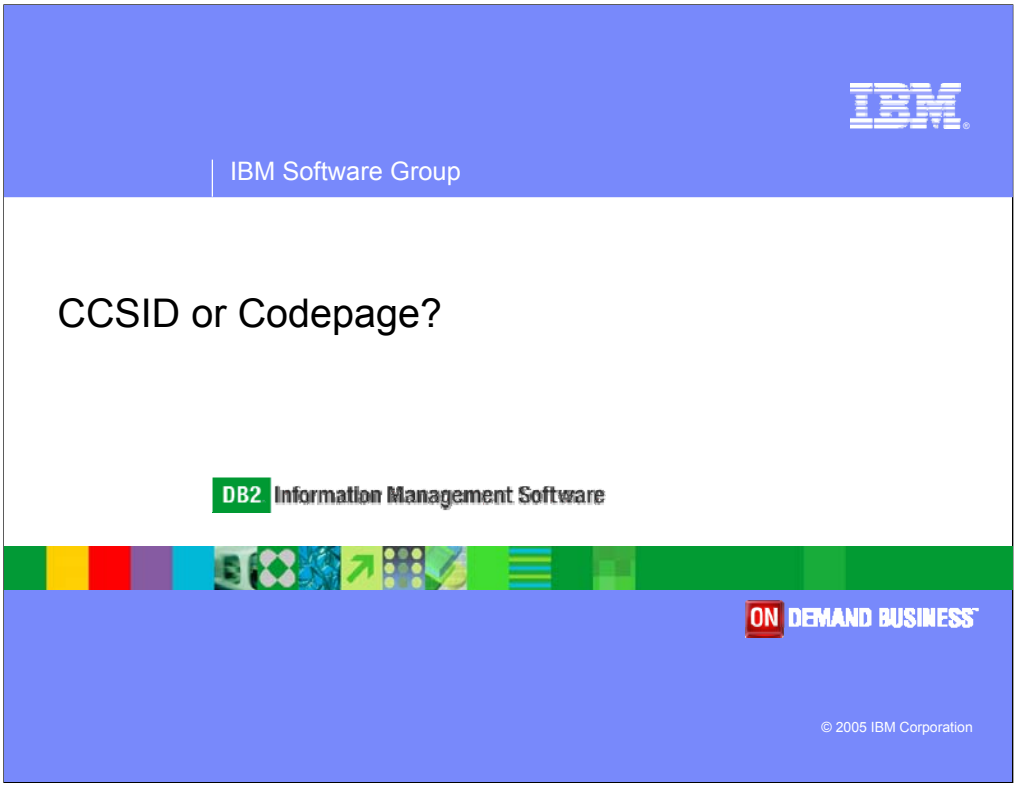

May 7–11, 2006
Tampa Convention Center
Tampa, Florida, USA
IDUG® 2006
North America
C01
CCSID 101 –
What's a CCSID and
why do I care?
Christopher J. Crone
IBM/DB2 for z/OS Development
Monday, May 8th, 2006 • 10:20 a.m. – 11:30 a.m.
Platform: z/OS
GoFurther
INTERNATIONAL
DB2 USERS GROUP

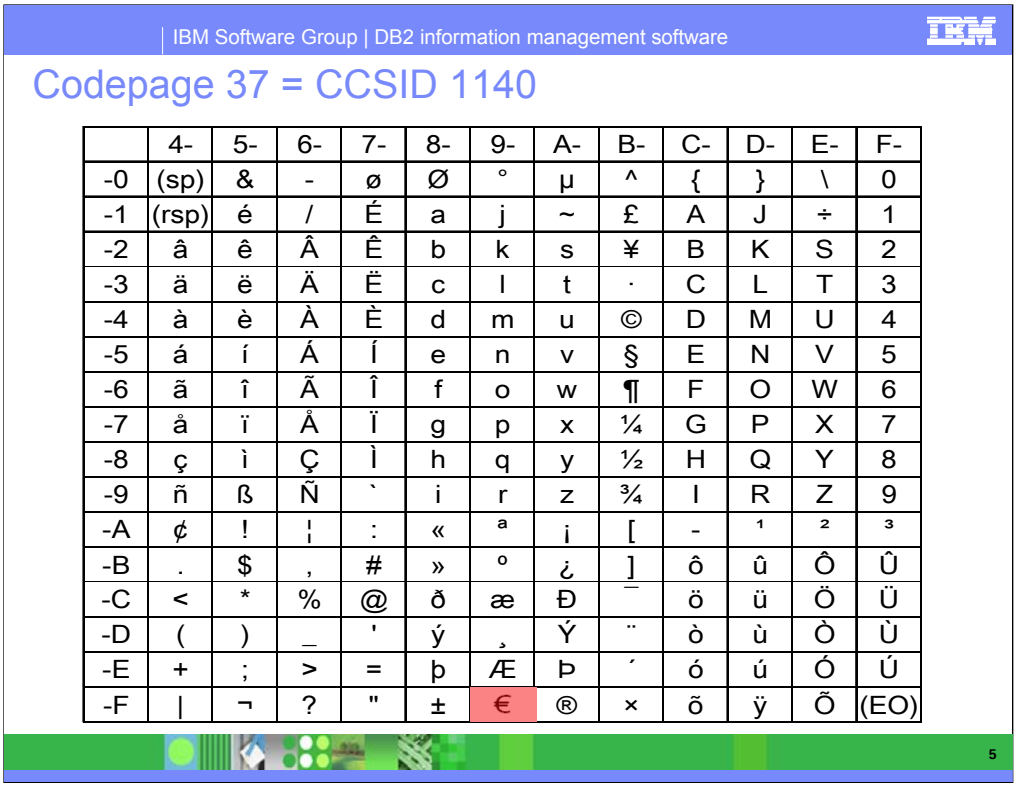
Presentation Topics

- CCSID or Codepage?
- ASCII, EBCDIC, and Unicode – Oh My!!!
- How is my data stored?
  - How can I tell?
- DRDA and DB2 Connect
- V7 –vs- V8
  - Main Differences

Terminology

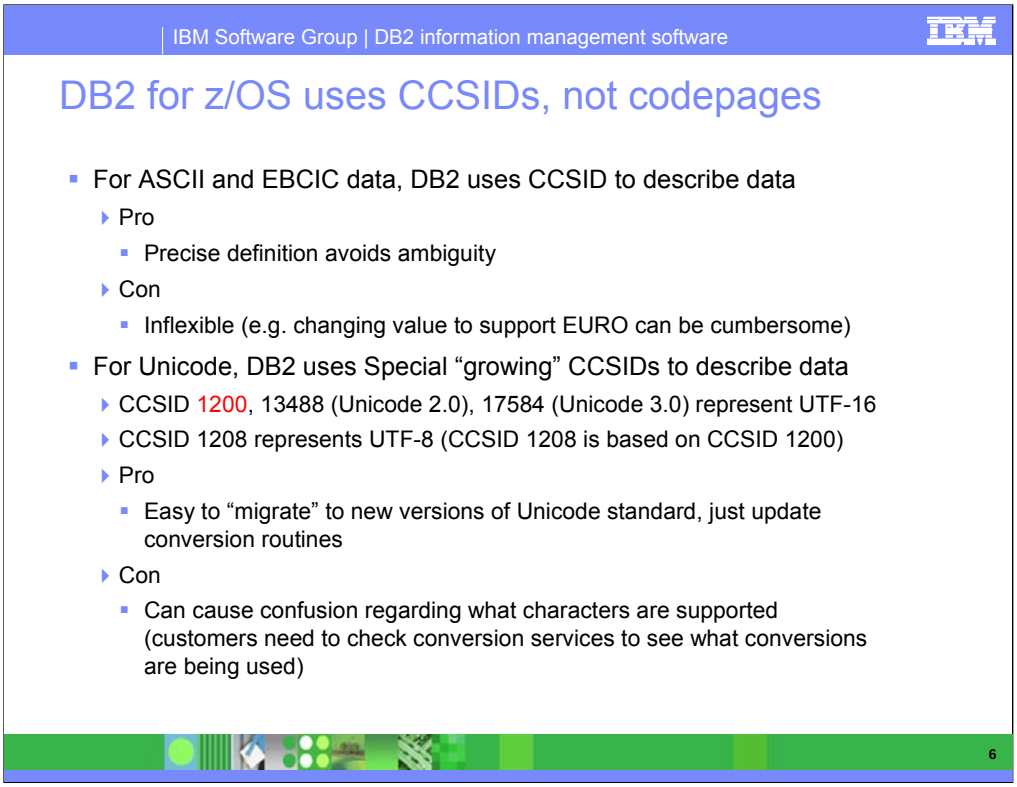
- For the purposes of this presentation
  - ASCII - all ASCII CCSIDs that DB2 currently supports
  - EBCDIC - all EBCDIC CCSIDs that DB2 currently supports
  - UNICODE - UTF-8 or UTF-16
  - Encoding Scheme
    - ASCII, EBCDIC or Unicode
  - CCSID – Coded Character Set Identifier **(used by DB2 to tag string data)**
    - Two-byte, unsigned binary integer identifying a specific set of encoding scheme and one or more pairs of character sets (CS) and code pages (CP)
  - CCSID set
    - The single byte CCSID value (SBCS), mixed CCSID value and double byte CCSID value (DBCS) associated with a particular encoding scheme
  - Multiple CCSID Sets
    - When two or more CCSID sets contain different CCSID values for one or more values in a set (SBCS, Mixed or DBCS).


IBM Software Group
CCSID or Codepage?
DB2 Information Management Software
ON DEMAND BUSINESS
© 2005 IBM Corporation



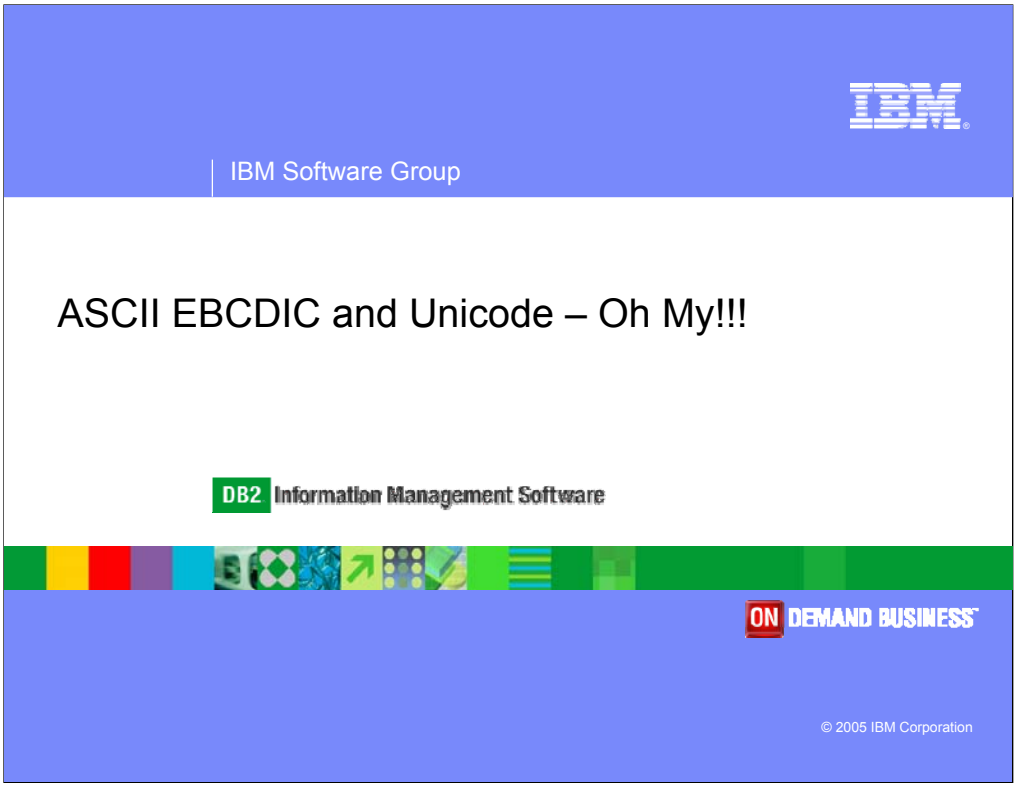
	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	^	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	¢	!	¦	:	«	ª	¡	[	-	¹	²	³
-B	.	$	,	#	»	º	¿	]	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	|	¬	?	"	±	€	®	×	õ	ÿ	Õ	(EO)

The words Codepage and CCSID are often used interchangeably. In some cases, these terms can be used interchangeably without harm. In the late 1990's, the Euro symbol was added to many EBCDIC codepages at x'9F'. In most cases the Euro replaced the International Currency Symbol (seen above at x'9F'). Once the Euro was added to Codepage 37, Codepage 37 and CCSID 37 diverged and can no longer be used interchangeably.



DB2 uses CCSIDs to describe data stored in DB2. In general this is good because DB2 can precisely describe your data. However, the negative is that it makes it hard to "change" your DB2 environment, for example to support the Euro symbol.

Unicode data uses Growing CCSIDs. This enables DB2 to represent your data in a precise way, yet still additions to the Unicode standard. The main drawback with this approach is that DB2 cannot tell which version of Unicode data is contained in Unicode tables. Since all Unicode conversions are performed using the z/OS conversion services, the level of Unicode supported by DB2 is really controlled by the level of the Unicode that the z/OS Unicode conversion services support.


IBM Software Group
ASCII EBCDIC and Unicode – Oh My!!!
DB2 Information Management Software
ON DEMAND BUSINESS
© 2005 IBM Corporation

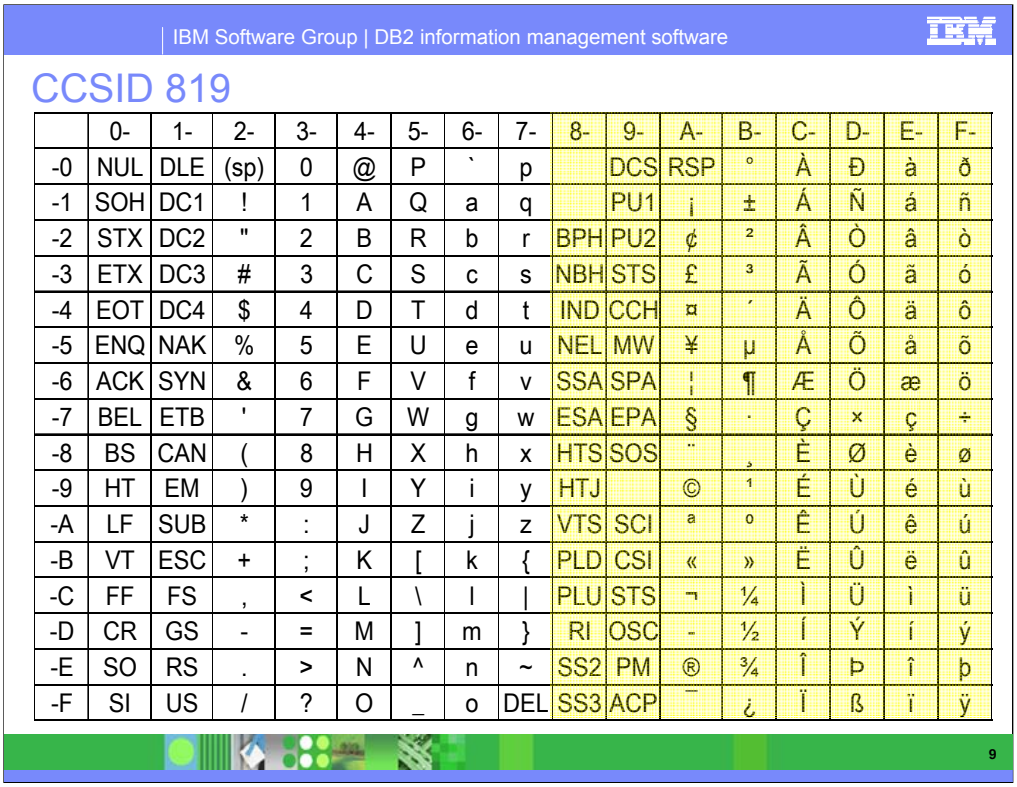
CCSID 367 – 7 Bit ASCII (Unicode SBCS data)

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	NUL	DLE	(sp)	0	@	P	`	p								
-1	SOH	DC1	!	1	A	Q	a	q								
-2	STX	DC2	"	2	B	R	b	r								
-3	ETX	DC3	#	3	C	S	c	s								
-4	EOT	DC4	$	4	D	T	d	t								
-5	ENQ	NAK	%	5	E	U	e	u								
-6	ACK	SYN	&	6	F	V	f	v								
-7	BEL	ETB	'	7	G	W	g	w								
-8	BS	CAN	(	8	H	X	h	x								
-9	HT	EM	)	9	I	Y	i	y								
-A	LF	SUB	*	:	J	Z	j	z								
-B	VT	ESC	+	;	K	[	k	{								
-C	FF	FS	,	<	L	\	l	|							
-D	CR	GS	-	=	M	]	m	}								
-E	SO	RS	.	>	N	^	n	~								
-F	SI	US	/	?	O	_	o									

CCSID 367 is 7 bit ASCII. It was often used in early ASCII computers (Unix systems). CCSID 367 is also used in “FOR SBCS DATA” columns in Unicode tables in DB2 for z/OS.

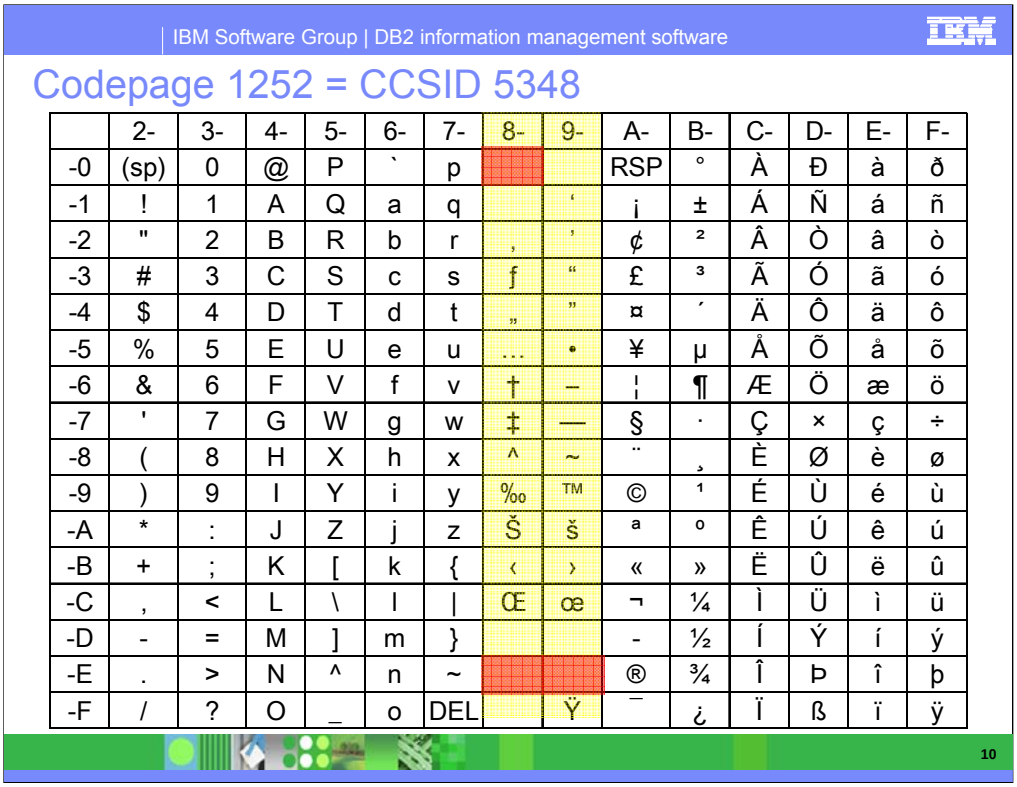
In ASCII, numbers are 1st, followed by uppercase numbers, and finally lowercase character.

Since this is a 7 bit ASCII CCSID, there are no characters mapped above x’7F’.



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	NUL	DLE	(sp)	0	@	P	`	p		DCS	RSP	°	À	Đ	à	ð
-1	SOH	DC1	!	1	A	Q	a	q		PU1	¡	±	Á	Ñ	á	ñ
-2	STX	DC2	"	2	B	R	b	r	BPH	PU2	¢	²	Â	Ò	â	ò
-3	ETX	DC3	#	3	C	S	c	s	NBH	STS	£	³	Ã	Ó	ã	ó
-4	EOT	DC4	$	4	D	T	d	t	IND	CCH	¤	´	Ä	Ô	ä	ô
-5	ENQ	NAK	%	5	E	U	e	u	NEL	MW	¥	µ	Å	Õ	å	õ
-6	ACK	SYN	&	6	F	V	f	v	SSA	SPA	¦	¶	Æ	Ö	æ	ö
-7	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	·	Ç	×	ç	÷
-8	BS	CAN	(	8	H	X	h	x	HTS	SOS	¨	¸	È	Ø	è	ø
-9	HT	EM	)	9	I	Y	i	y	HTJ		©	¹	É	Ù	é	ù
-A	LF	SUB	*	:	J	Z	j	z	VTS	SCI	ª	º	Ê	Ú	ê	ú
-B	VT	ESC	+	;	K	[	k	{	PLD	CSI	«	»	Ë	Û	ë	û
-C	FF	FS	,	<	L	\	l	|	PLU	STS	¬	¼	Ì	Ü	ì	ü
-D	CR	GS	-	=	M	]	m	}	RI	OSC	-	½	Í	Ý	í	ý
-E	SO	RS	.	>	N	^	n	~	SS2	PM	®	¾	Î	Þ	î	þ
-F	SI	US	/	?	O	_	o	DEL	SS3	ACP	¯	¿	Ï	ß	ï	ÿ

CCSID 819 is a superset of CCSID 367 – note that CCSID 819 has support for accented characters, and other extended characters such as AE ligature. CCSID 819 is a common ASCII CCSID used in AIX, SUN Solaris, and Linux.

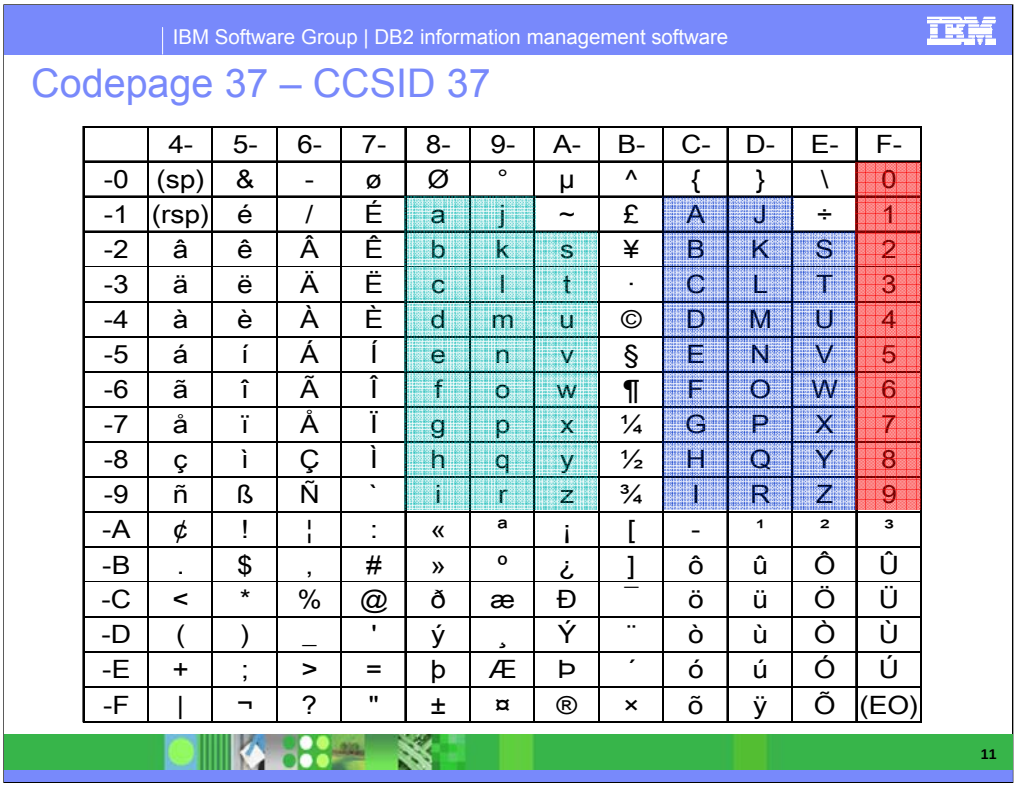


	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	0	@	P	`	p			RSP	°	À	Đ	à	ð
-1	!	1	A	Q	a	q		‘	¡	±	Á	Ñ	á	ñ
-2	"	2	B	R	b	r	‚	’	¢	²	Â	Ò	â	ò
-3	#	3	C	S	c	s	ƒ	“	£	³	Ã	Ó	ã	ó
-4	$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
-5	%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å	õ
-6	&	6	F	V	f	v	†	–	¦	¶	Æ	Ö	æ	ö
-7	'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
-8	(	8	H	X	h	x	^	~	¨	¸	È	Ø	è	ø
-9	)	9	I	Y	i	y	‰	™	©	¹	É	Ù	é	ù
-A	*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
-B	+	;	K	[	k	{	‹	›	«	»	Ë	Û	ë	û
-C	,	<	L	\	l	|	Œ	œ	¬	¼	Ì	Ü	ì	ü
-D	-	=	M	]	m	}			-	½	Í	Ý	í	ý
-E	.	>	N	^	n	~			®	¾	Î	Þ	î	þ
-F	/	?	O	_	o	DEL		Ÿ	¯	¿	Ï	ß	ï	ÿ

CCSID 5348 is Codepage 1252 with Euro support. It is commonly associated with Microsoft Windows™.

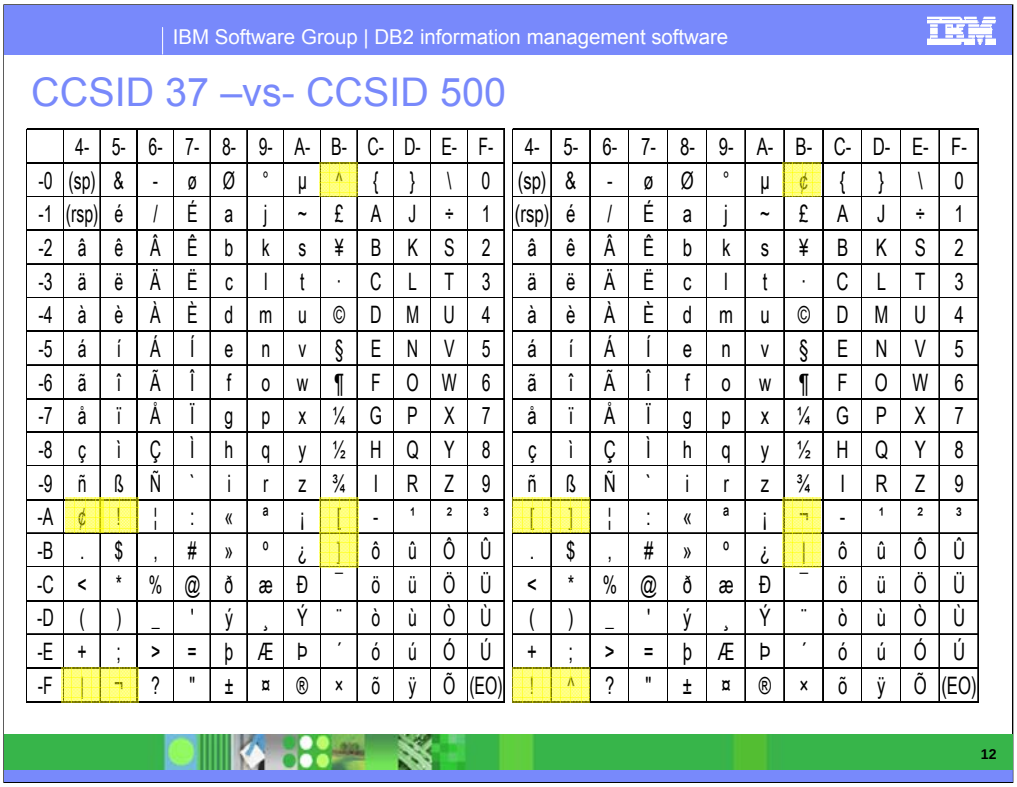
From the perspective of displayable characters, CCSID 5348 is a superset of CCSID 819. CCSID 5348 manages this by replacing the control characters between x’80’ – x’9F’ that exist in CCSID 819, with additional characters such as the ™ symbol or the œ ligature.

CCSID 5348 maps additional characters, such as the ™ character, at the expense of control characters.



	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	^	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	¢	!	¦	:	«	ª	¡	[	-	¹	²	³
-B	.	$	,	#	»	º	¿	]	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	|	¬	?	"	±	¤	®	×	õ	ÿ	Õ	(EO)

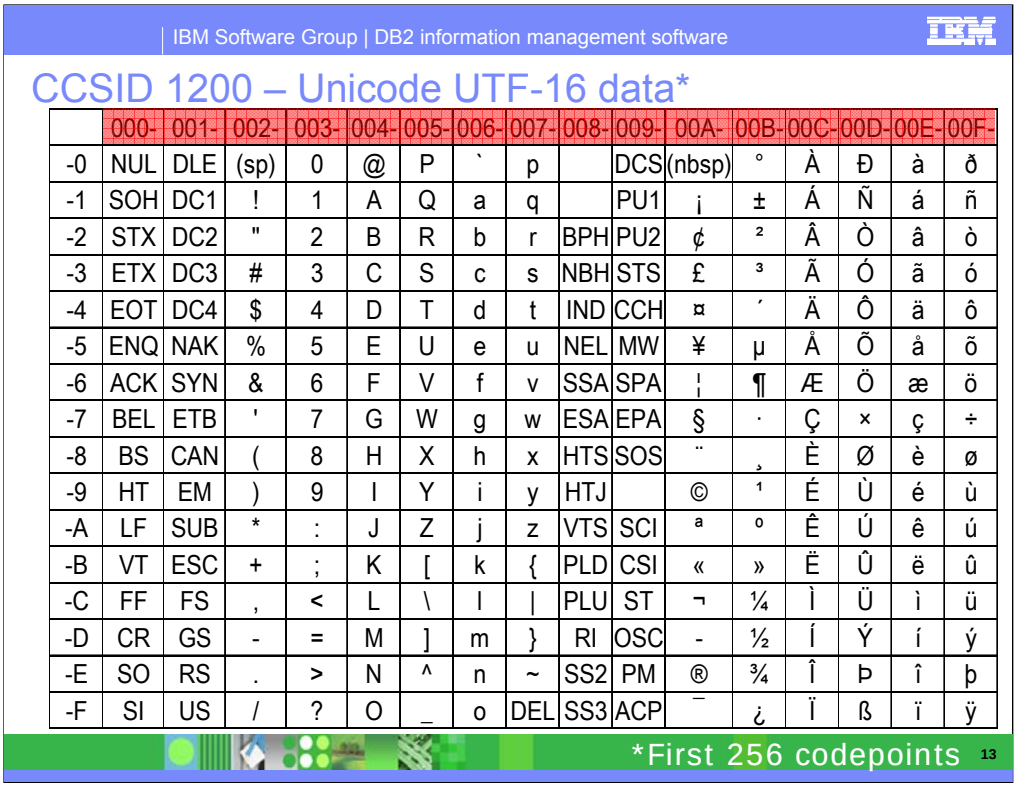
In EBCDIC, lowercase characters come 1st, then uppercase characters, and finally numbers. The codepoints used to map a-z and A-Z are not contiguous (as they are in ASCII).



	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	^	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	¢	!	¦	:	«	ª	¡	[	-	¹	²	³
-B	.	$	,	#	»	º	¿	]	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	|	¬	?	"	±	¤	®	×	õ	ÿ	Õ	(EO)

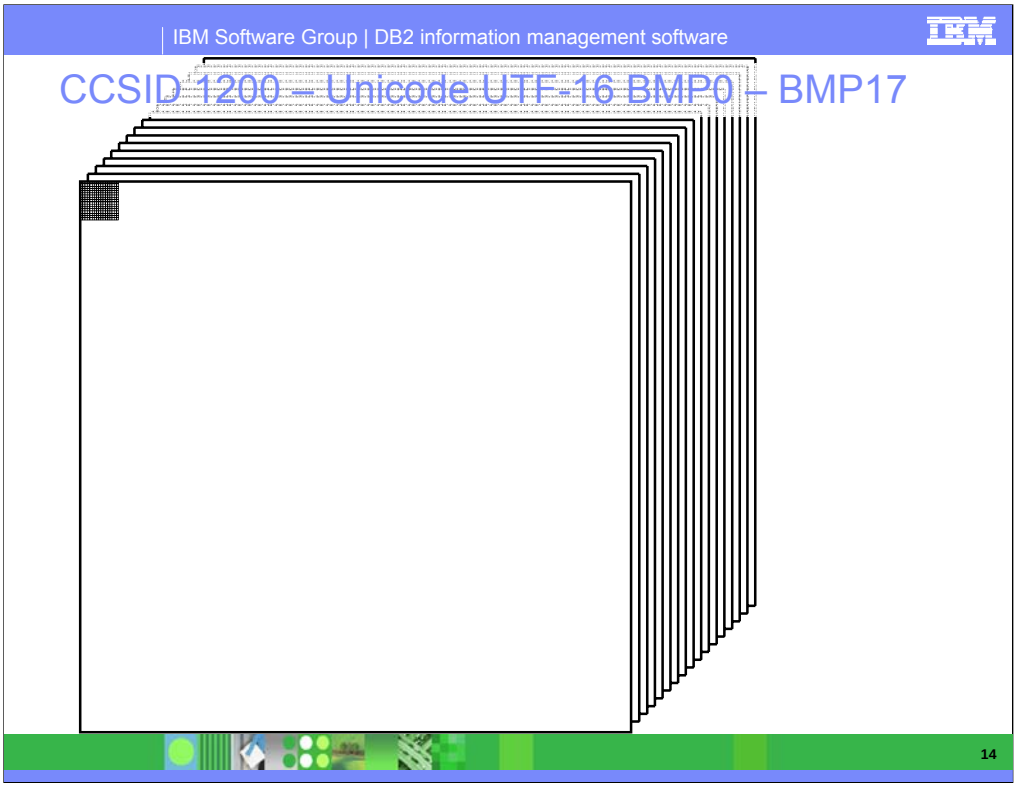
	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	¢	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	[	]	¦	:	«	ª	¡	¬	-	¹	²	³
-B	.	$	,	#	»	º	¿	|	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	¦	^	?	"	±	¤	®	×	õ	ÿ	Õ	(EO)

There are Seven Characters that are different between CCSID 37 and CCSID 500.



	000-	001-	002-	003-	004-	005-	006-	007-	008-	009-	00A-	00B-	00C-	00D-	00E-	00F-
-0	NUL	DLE	(sp)	0	@	P	`	p		DCS	(nbsp)	°	À	Đ	à	ð
-1	SOH	DC1	!	1	A	Q	a	q		PU1	¡	±	Á	Ñ	á	ñ
-2	STX	DC2	"	2	B	R	b	r	BPH	PU2	¢	²	Â	Ò	â	ò
-3	ETX	DC3	#	3	C	S	c	s	NBH	STS	£	³	Ã	Ó	ã	ó
-4	EOT	DC4	$	4	D	T	d	t	IND	CCH	¤	´	Ä	Ô	ä	ô
-5	ENQ	NAK	%	5	E	U	e	u	NEL	MW	¥	µ	Å	Õ	å	õ
-6	ACK	SYN	&	6	F	V	f	v	SSA	SPA	¦	¶	Æ	Ö	æ	ö
-7	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	·	Ç	×	ç	÷
-8	BS	CAN	(	8	H	X	h	x	HTS	SOS	¨	¸	È	Ø	è	ø
-9	HT	EM	)	9	I	Y	i	y	HTJ		©	¹	É	Ù	é	ù
-A	LF	SUB	*	:	J	Z	j	z	VTS	SCI	ª	º	Ê	Ú	ê	ú
-B	VT	ESC	+	;	K	[	k	{	PLD	CSI	«	»	Ë	Û	ë	û
-C	FF	FS	,	<	L	\	l	|	PLU	ST	¬	¼	Ì	Ü	ì	ü
-D	CR	GS	-	=	M	]	m	}	RI	OSC	-	½	Í	Ý	í	ý
-E	SO	RS	.	>	N	^	n	~	SS2	PM	®	¾	Î	Þ	î	þ
-F	SI	US	/	?	O	_	o	DEL	SS3	ACP	¯	¿	Ï	ß	ï	ÿ

The first 256 codepoints of UTF-16 are the same as CCSID 819. Note that the codepoints from 0080-009F do not map characters (like they do in CCSIDs 1252 or 5348).



Unicode UTF-16 is composed of BMP0 (Basic Multilingual Plane 0), and 16 additional planes, BMP1 – BMP17. Each plane is capable of representing 66536 characters for a total of 1,114,112 characters.

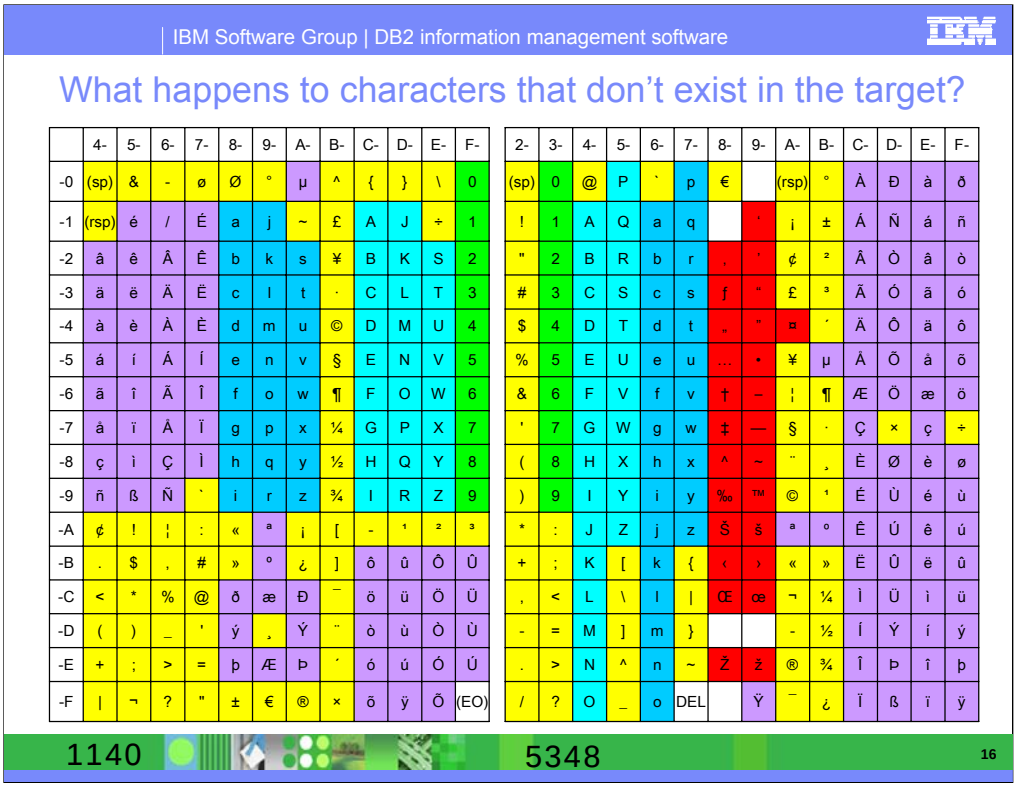
The little square in the upper left hand corner represents the first 256 codepoints of UTF-16 (this picture is to scale). As you can see, the repertoire of characters that Unicode is able to represent is much larger (over 4000 times) than a typical 256 character ASCII or EBCDIC CCSID.

Character examples

Character	ASCII	UTF-8	UTF-16 (Big Endian format)	UTF-32 (Big Endian format)
A	'41'x	'41'x	'0041'x	'00000041'x
a	'61'x	'61'x	'0061'x	'00000061'x
9	'39'x	'39'x	'0039'x	'00000039'x
Å (The character A with Ring accent)	'C5'x	'C385'x Note: 'C5'x becomes double byte in UTF-8	'00C5'x	'000000C5'x
顯 U+9860	'CDDB'x (CCSID 939)	'E9A1A0'x	'9860'x	'00009860'x
𠀐 U+200D0	N/A	'F0A08390'x	'D840DCD0'x	'000200D0'x

Note: UCS-2/UTF-16 and UCS-4/UTF-32 are using a technique called Zero Extension

This table shows some characters and their representations in different encodings.
Notice that **Å** changes from 1 byte in ASCII to 2 bytes in UTF-8 or UTF-16
Notice that the sizes and representations are different in UTF-8 and UTF-16



	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	^	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	¢	!	¦	:	«	ª	¡	[	-	¹	²	³
-B	.	$	,	#	»	º	¿	]	õ	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	|	¬	?	"	±	€	®	×	õ	ÿ	Õ	(EO)

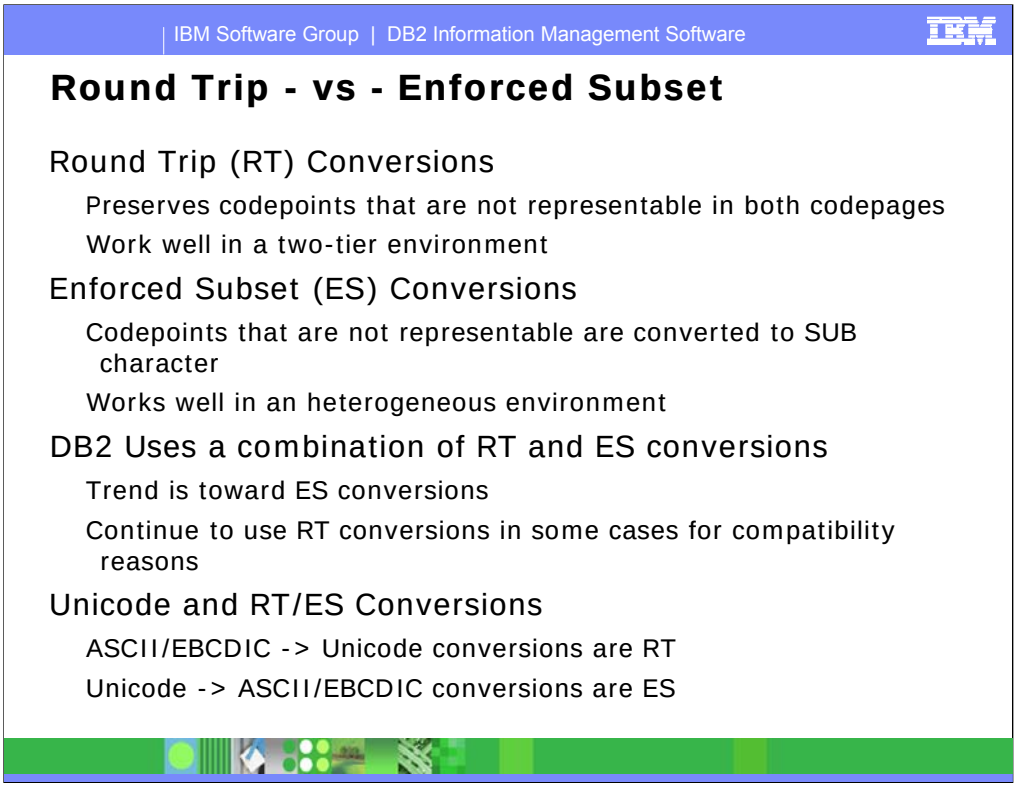
	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	0	@	P	`	p	€		(rsp)	°	À	Ð	à	ð
-1	!	1	A	Q	a	q		‘	¡	±	Á	Ñ	á	ñ
-2	"	2	B	R	b	r	‚	’	¢	²	Â	Ò	â	ò
-3	#	3	C	S	c	s	ƒ	“	£	³	Ã	Ó	ã	ó
-4	$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
-5	%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å	õ
-6	&	6	F	V	f	v	†	–	¦	¶	Æ	Ö	æ	ö
-7	'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
-8	(	8	H	X	h	x	^	~	¨	¸	È	Ø	è	ø
-9	)	9	I	Y	i	y	‰	™	©	¹	É	Ù	é	ù
-A	*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
-B	+	;	K	[	k	{	‹	›	«	»	Ë	Û	ë	û
-C	,	<	L	\	l	|	Œ	œ	¬	¼	Ì	Ü	ì	ü
-D	-	=	M	]	m	}			-	½	Í	Ý	í	ý
-E	.	>	N	^	n	~	Ž	ž	®	¾	Î	Þ	î	þ
-F	/	?	O	_	o	DEL		Ÿ	¯	¿	Ï	ß	ï	ÿ

Conversion results can be different for Round Trip, or Enforced Subset Conversions.

On the left we have EBCDIC CCSID 1140, on the right we have ASCII CCSID 5348. Characters in Blue, Cyan, Green, and Yellow all map directly from CCSID 5348 to CCSID 1140. The characters in Red do not map to corresponding characters on CCSID 1140.

When using a Round Trip (RT) conversion, the characters in Red map to control characters. They are not available to EBCDIC based applications (for instance, you could not view them in SPUFI), however when the data is converted back to ASCII (5348), the characters are returned to their original representation.

When using an Enforced Subset (ES) conversion, the characters in Red are converted to the SUB character (x'3F' in EBCDIC). The data is lost at this point and cannot be restored to it's original representation.



DB2 conversions are either Round Trip, or Enforced Subset.

Round Trip conversions attempt to avoid loss of data by mapping unrepresented codepoints to unused (or unlikely to be used) codepoints

-Data loss can be avoided or delayed

-Unpredictable conversions can occur in heterogeneous environments

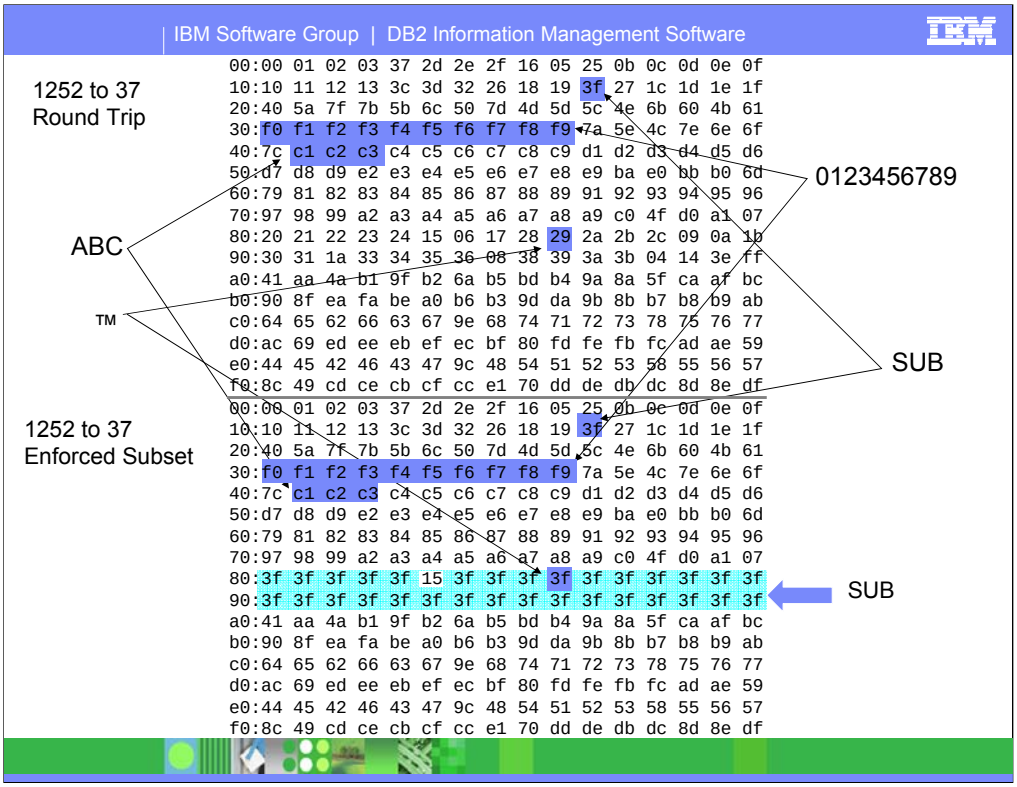
Enforced Subset conversions map any unrepresented codepoints to the sub-character

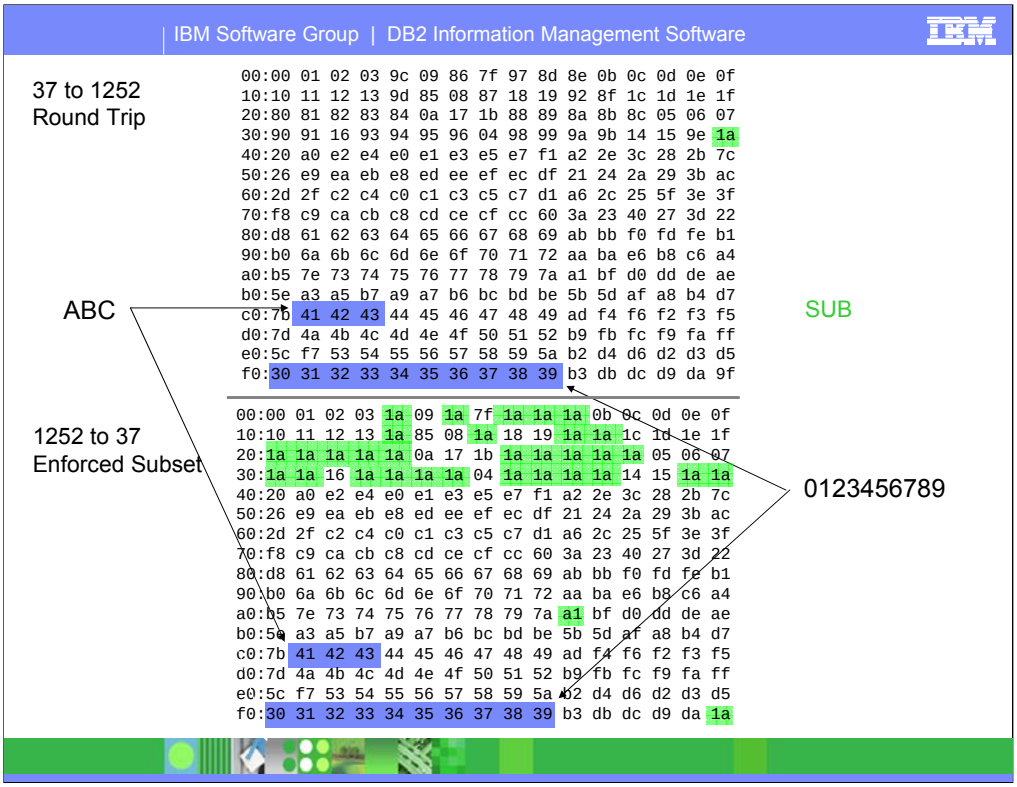
-Data loss occurs immediately in a predictable manner

Conversions to Unicode are RT conversions

Conversions to ASCII/EBCDIC from Unicode are ES

-Unicode is capable of representing many more characters than ASCII or EBCDIC.


IBM Software Group | DB2 Information Management Software
1252 to 37
Round Trip
00:00 01 02 03 37 2d 2e 2f 16 05 25 0b 0c 0d 0e 0f
10:10 11 12 13 3c 3d 32 26 18 19 3f 27 1c 1d 1e 1f
20:40 5a 7f 7b 5b 6c 50 7d 4d 5d 5c 4e 6b 60 4b 61
30:f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 7a 5e 4c 7e 6e 6f
40:7c c1 c2 c3 c4 c5 c6 c7 c8 c9 d1 d2 d3 d4 d5 d6
50:d7 d8 d9 e2 e3 e4 e5 e6 e7 e8 e9 ba e0 bb b0 6d
60:79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96
70:97 98 99 a2 a3 a4 a5 a6 a7 a8 a9 c0 4f d0 a1 07
80:20 21 22 23 24 15 06 17 28 29 2a 2b 2c 09 0a 1b
90:30 31 1a 33 34 35 36 08 38 39 3a 3b 04 14 3e ff
a0:41 aa 4a b1 9f b2 6a b5 bd b4 9a 8a 5f ca af bc
b0:90 8f ea fa be a0 b6 b3 9d da 9b 8b b7 b8 b9 ab
c0:64 65 62 66 63 67 9e 68 74 71 72 73 78 75 76 77
d0:ac 69 ed ee eb ef ec bf 80 fd fe fb fc ad ae 59
e0:44 45 42 46 43 47 9c 48 54 51 52 53 58 55 56 57
f0:8c 49 cd ce cb cf cc e1 70 dd de db dc 8d 8e df
0123456789
ABC
TM
SUB
1252 to 37
Enforced Subset
00:00 01 02 03 37 2d 2e 2f 16 05 25 0b 0c 0d 0e 0f
10:10 11 12 13 3c 3d 32 26 18 19 3f 27 1c 1d 1e 1f
20:40 5a 7f 7b 5b 6c 50 7d 4d 5d 5c 4e 6b 60 4b 61
30:f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 7a 5e 4c 7e 6e 6f
40:7c c1 c2 c3 c4 c5 c6 c7 c8 c9 d1 d2 d3 d4 d5 d6
50:d7 d8 d9 e2 e3 e4 e5 e6 e7 e8 e9 ba e0 bb b0 6d
60:79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96
70:97 98 99 a2 a3 a4 a5 a6 a7 a8 a9 c0 4f d0 a1 07
80:3f 3f 3f 3f 3f 15 3f 3f 3f 3f 3f 3f 3f 3f 3f 3f
90:3f 3f 3f 3f 3f 3f 3f 3f 3f 3f 3f 3f 3f 3f 3f 3f
a0:41 aa 4a b1 9f b2 6a b5 bd b4 9a 8a 5f ca af bc
b0:90 8f ea fa be a0 b6 b3 9d da 9b 8b b7 b8 b9 ab
c0:64 65 62 66 63 67 9e 68 74 71 72 73 78 75 76 77
d0:ac 69 ed ee eb ef ec bf 80 fd fe fb fc ad ae 59
e0:44 45 42 46 43 47 9c 48 54 51 52 53 58 55 56 57
f0:8c 49 cd ce cb cf cc e1 70 dd de db dc 8d 8e df
SUB


IBM Software Group | DB2 Information Management Software
37 to 1252
Round Trip
00:00 01 02 03 9c 09 86 7f 97 8d 8e 0b 0c 0d 0e 0f
10:10 11 12 13 9d 85 08 87 18 19 92 8f 1c 1d 1e 1f
20:80 81 82 83 84 0a 17 1b 88 89 8a 8b 8c 05 06 07
30:90 91 16 93 94 95 96 04 98 99 9a 9b 14 15 9e 1a
40:20 a0 e2 e4 e0 e1 e3 e5 e7 f1 a2 2e 3c 28 2b 7c
50:26 e9 ea eb e8 ed ee ef ec df 21 24 2a 29 3b ac
60:2d 2f c2 c4 c0 c1 c3 c5 c7 d1 a6 2c 25 5f 3e 3f
70:f8 c9 ca cb c8 cd ce cf cc 60 3a 23 40 27 3d 22
80:d8 61 62 63 64 65 66 67 68 69 ab bb f0 fd fe b1
90:b0 6a 6b 6c 6d 6e 6f 70 71 72 aa ba e6 b8 c6 a4
a0:b5 7e 73 74 75 76 77 78 79 7a a1 bf d0 dd de ae
b0:5e a3 a5 b7 a9 a7 b6 bc bd be 5b 5d af a8 b4 d7
c0:7b 41 42 43 44 45 46 47 48 49 ad f4 f6 f2 f3 f5
d0:7d 4a 4b 4c 4d 4e 4f 50 51 52 b9 fb fc f9 fa ff
e0:5c f7 53 54 55 56 57 58 59 5a b2 d4 d6 d2 d3 d5
f0:30 31 32 33 34 35 36 37 38 39 b3 db dc d9 da 9f
ABC
SUB
1252 to 37
Enforced Subset
00:00 01 02 03 1a 09 1a 7f 1a 1a 1a 0b 0c 0d 0e 0f
10:10 11 12 13 1a 85 08 1a 18 19 1a 1a 1c 1d 1e 1f
20:1a 1a 1a 1a 1a 0a 17 1b 1a 1a 1a 1a 1a 05 06 07
30:1a 1a 16 1a 1a 1a 1a 04 1a 1a 1a 1a 14 15 1a 1a
40:20 a0 e2 e4 e0 e1 e3 e5 e7 f1 a2 2e 3c 28 2b 7c
50:26 e9 ea eb e8 ed ee ef ec df 21 24 2a 29 3b ac
60:2d 2f c2 c4 c0 c1 c3 c5 c7 d1 a6 2c 25 5f 3e 3f
70:f8 c9 ca cb c8 cd ce cf cc 60 3a 23 40 27 3d 22
80:d8 61 62 63 64 65 66 67 68 69 ab bb f0 fd fe b1
90:b0 6a 6b 6c 6d 6e 6f 70 71 72 aa ba e6 b8 c6 a4
a0:b5 7e 73 74 75 76 77 78 79 7a a1 bf d0 dd de ae
b0:5e a3 a5 b7 a9 a7 b6 bc bd be 5b 5d af a8 b4 d7
c0:7b 41 42 43 44 45 46 47 48 49 ad f4 f6 f2 f3 f5
d0:7d 4a 4b 4c 4d 4e 4f 50 51 52 b9 fb fc f9 fa ff
e0:5c f7 53 54 55 56 57 58 59 5a b2 d4 d6 d2 d3 d5
f0:30 31 32 33 34 35 36 37 38 39 b3 db dc d9 da 1a
0123456789

Endianess

Big Endian
- pSeries, zSeries, iSeries, Sun, HP
- Most significant byte is leftmost
  - For a 4 byte word - Byte order 0,1,2,3

Little Endian
- Intel based machines including xSeries
- Least significant byte is leftmost
  - For a 4 byte word - Byte order 3,2,1,0

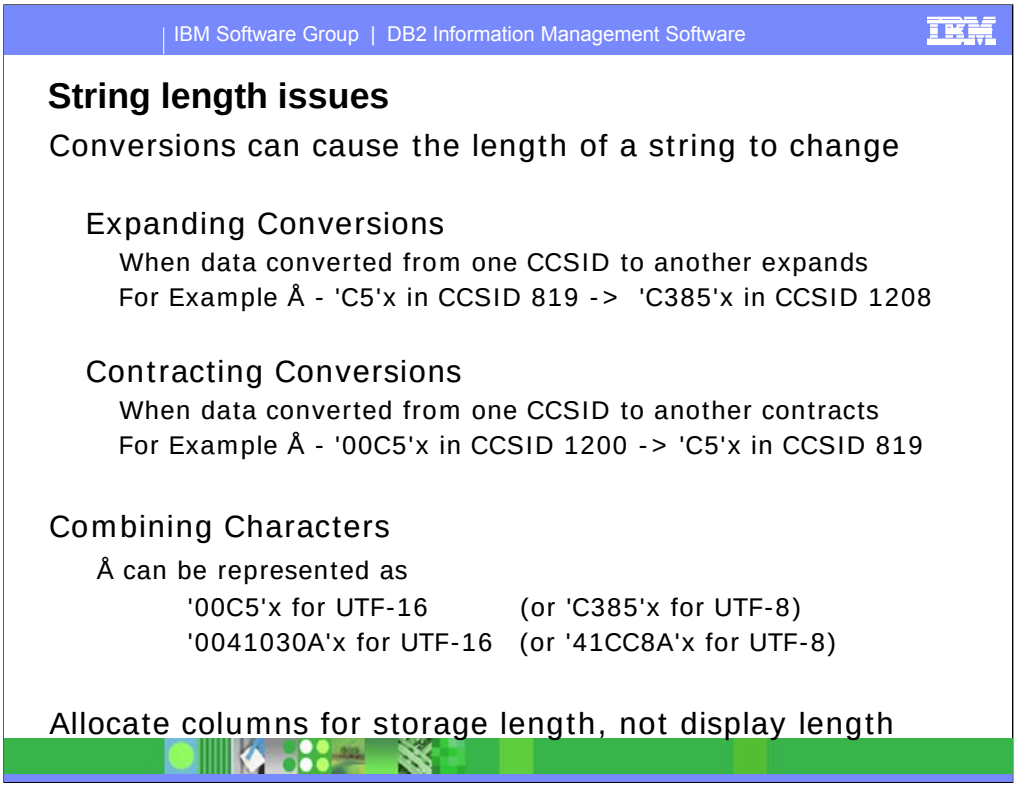
UTF-8 - not affected by endianess issues

UTF-16 and UTF-32 are effected by endianess issues
- Big Endian
  - 'A' = x'0041' for UTF-16 or x'00000041' for UTF-32
- Little Endian
  - 'A' = x'4100' for UTF-16 or x'41000000' for UTF-32

Note: A BYTE is always ordered as leftmost most significant bit to rightmost least significant bit. Bit order within a byte is always 7,6,5,4,3,2,1,0

Unicode UTF-16 data is represented as an Unsigned HW
Unicode UTF-32 data is represented as an Unsigned FW
HWs and FWs have an Endianess associated with them.
DB2 and DRDA manipulate and store data in Big Endian format.

Little Endian clients convert data to Big Endian before putting on the wire.



There are some cases where a conversion causes the length of the data to change.
-Expanding conversions cause the length of the data to grow
-Contracting conversions cause the length of the data to shrink
Combining Characters in Unicode can also cause length issue
Character datatypes in DB2 are byte oriented – not NLS character oriented.

Table 25. "Result length of CCSID conversion" in Chapter 2 of the describes the DB2 rules for expanding and contracting conversions.

Conversion methods

Native DB2
- SYSIBM.SYSSTRINGS (V2.3)

OS Conversion services
- ICONV (Requires OS/390 V2R9 and above)
  - Uses LE base services (V7 only)
- **z/OS support for Unicode (V7 & V8)**
  - **31 and 64 bit capable (after z/OS V1R3).**

DB2 uses three methods for conversion
-SYSSTRINGS - This is the conversion services that were introduced in DB2 V2R3 and the ones most people are familiar with
-ICONV - Introduced in DB2 V6 – discontinued with DB2 V8 .
- -This was our first attempt at leveraging OS/390 infrastructure to perform conversion.
- -It is non-strategic because the OS/390 V2 R8/R9/R10 support for Unicode provides more functionality with better performance

-OS/390 V2 R8/R9/R10 support for Unicode - Starting in V7, DB2 will be leveraging this service for most future character conversion support.

Conversion services configuration

Which Conversions should be configured

- CCSID 367 (7-bit ASCII) <-> ASCII & EBCDIC System CCSID(s)
- CCSID 1208 (UTF-8) <-> ASCII & EBCDIC System CCSID(s)
- CCSID 1200 (UTF-16) <-> ASCII & EBCDIC System CCSID(s)
- Client CCSID(s) <-> Unicode CCSIDs (367, 1208, 1200)
- CCSID 37,500, and 1047 <-> Unicode CCSIDs (367, 1208, 1200)
- ASCII or EBCDIC Conversions not included in SYSSTRINGS
- Other
  - Conversions needed to LOAD/UNLOAD Data
  - Conversions needed to support application encoding bind option, DECLARE VARIABLE, or CCSID overrides

II13048 and II13049 have detailed information

See example in Reference section in back of presentation

Which Conversions should be configured
-All DB2 conversions involving Unicode are supported via the OS/390 conversion services. Any conversion involving Unicode must be configured
-ASCII and EBCDIC conversions not supported by Native DB2 Conversion methods
-Other conversions that you might need to support DRDA clients, LOAD/UNLOAD, or Application directed conversions.
-New with V8, you must have conversions to/from Unicode configured for CCSIDs 37, 500, and 1047. These conversions are needed by DB2 for internal processing.

z/OS 1.4 +APARS
Generic Conversion Support and Changes in Display output

DISPLAY UNI issuers will now find possibly more conversions listed on the console output since now each of the techniques will be loaded in storage or in the image.

Example: `CONVERSION 37,850,RECL`
Before:

```
CONVERSION: 00037-00850-RECL
```

After:

```
CONVERSION: 00037-00850-R                00037-00850-E
            00037-00850-C                00037-00850-L
```

Note: Tables for conversions listed must be available on the system z/OS 1.4 and higher, via APAR OA08723. You will also need APAR OA10699.

z/OS 1.7 - Unicode on Demand

- Eliminates the need to ever customize Unicode Services
  - Customers, no longer have to run image generator program (to build all the tables they need into an “image”) and customize the CUNUNIxx PARMLIB member with this image name, in order for tables to be loaded into storage.
  - All tables needed for a character conversion, case conversion, normalization and collation services will now be loaded once they are required and not present in storage.
  - The caller requests of Unicode services will no longer fail except when invalid or unsupported conversions or service are specified.
  - The necessary tables will be loaded dynamically under the covers to satisfy the caller request.
  - Image no longer needs to be placed in PARMLIB data set concatenation.
  - Image generator program will still be shipped and existing images and new images will still be supported.

- New Operator command
  - **SETUNI ADD**
    - SETUNI ADD,FROM=xxxx,TO=yyyy [,TECHNIQUE=] [,DSNAME=] [,VOLSER=]
    - SETUNI ADD,CASE [=LOCAL | [,SPECIAL] | [,NORMAL]] [,DSNAME=] [,VOLSER=]
    - SETUNI ADD,NORMALIZE [,DSNAME=] [,VOLSER=]
    - SETUNI ADD,COLLATE [,DSNAME=] [,VOLSER=]
    - SETUNI ADD,IMAGE=zzzzzzzz [,DSNAME=] [,VOLSER=]
      - To add an image, especially allowing for it to be loaded from other than PARMLIB concatenation. When this is specified, we will analyze the tables in the image specified and compare them to the tables in storage. Intersecting tables will not be added.

- New Operator command *(cont.)*

  - **SETUNI DELETE**
    - SETUNI DELETE,FROM=xxxx,TO=yyyy [,TECHNIQUE=],FORCE=YES
    - SETUNI DELETE,CASE [=LOCAL | [,SPECIAL] | [,NORMAL]],FORCE=YES
    - SETUNI DELETE,NORMALIZE,FORCE=YES
    - SETUNI DELETE,COLLATE,FORCE=YES
      - FORCE=YES must be specified, to ensure the customer realizes what they are doing, since we have no idea which applications could still be using the handle (pointer) to the tables and the storage occupied by the tables will be freed.
      - Note: FORCE=NO is not valid

New Operator command *(cont.)*

- **SETUNI REPLACE -** To replace individual tables that may currently be in storage. When REPLACE is specified but the tables to be replaced are not in the Unicode Environment, a simple ADD operation is performed.
  - SETUNI REPLACE,FROM=xxxx,TO=yyyy,[TECHNIQUE=],[DSNAME=] [,VOLSER=] [,FREE=NO | (YES,FORCE)]
  - SETUNI REPLACE,CASE [=LOCAL | [,SPECIAL] | [,NORMAL]] [,DSNAME=] [,VOLSER=] [,FREE=NO | (YES,FORCE)]
  - SETUNI REPLACE,NORMALIZE [,DSNAME=] [,VOLSER=] [,FREE=NO | (YES,FORCE)]
  - SETUNI REPLACE,COLLATE [,DSNAME=] [,VOLSER=] [,FREE=NO | (YES,FORCE)]
  - Note: The FREE keyword is optional, denoting whether or not we want to release the storage associated with this table. (YES,FORCE) will release the storage associated with that table. This is like doing a “defrag” of our dataspace. NO will not release the storage. When FREE is not specified, we will not release storage.

- New Operator command *(cont.)*
  - **SETUNI REALSTORAGE**
    - SETUNI REALSTORAGE=nnnnn (in pages)
    - SETUNI REALSTORAGE=xxG (in Gigabytes)
    - SETUNI REALSTORAGE=xxM (in Megabytes)
    - SETUNI REALSTORAGE=xxK (in Kilobytes)
      - Note: DEFAULT value is 2Gb

Result length of CCSID conversion

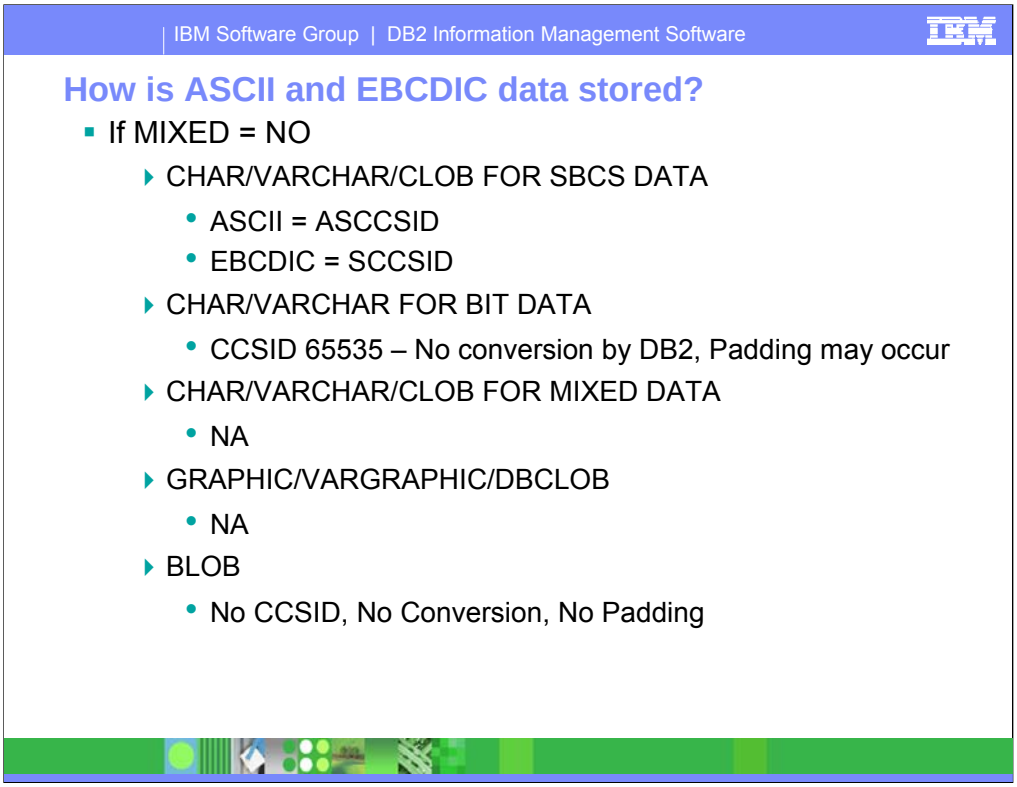
		To CCSID								
		EBCDIC			ASCII			Unicode		
From CCSID		SBCS	Mixed	DBCS	SBCS	Mixed	DBCS	SBCS	UTF-8	UTF-16
EBCDIC	SBCS	X	X	X*2[1]	X	X	X*2[1]	X[1]	X*3	X*2
	Mixed	X	X	X*2[1]	X	X	X*2[1]	X[1]	X*3	X*2
	DBCS	X*0.5[1]	X+2	X	X*0.5[1]	X	X	X*0.5	X*1.5	X
ASCII	SBCS	X	X	X*2[1]	X	X	X*2[1]	X[1]	X*3	X*2
	Mixed	X	X*1.8	X*2[1]	X	X	X*2[1]	X[1]	X*3	X*2
	DBCS	X*0.5[1]	X+2	X	X*0.5[1]	X	X	X*0.5	X*1.5	X
Unicode	SBCS	X	X	X*2	X	X	X*2	X	X	X*2
	UTF-8	X	X*1.25	X	X	X	X	X	X	X*2
	UTF-16	X*0.5	X+2	X	X*0.5	X	X	X*0.5	X*1.5	X

Notes:

1. Because of the high probability of data loss, IBM does not provide conversion tables for this combination of two CCSIDs and data subtypes.

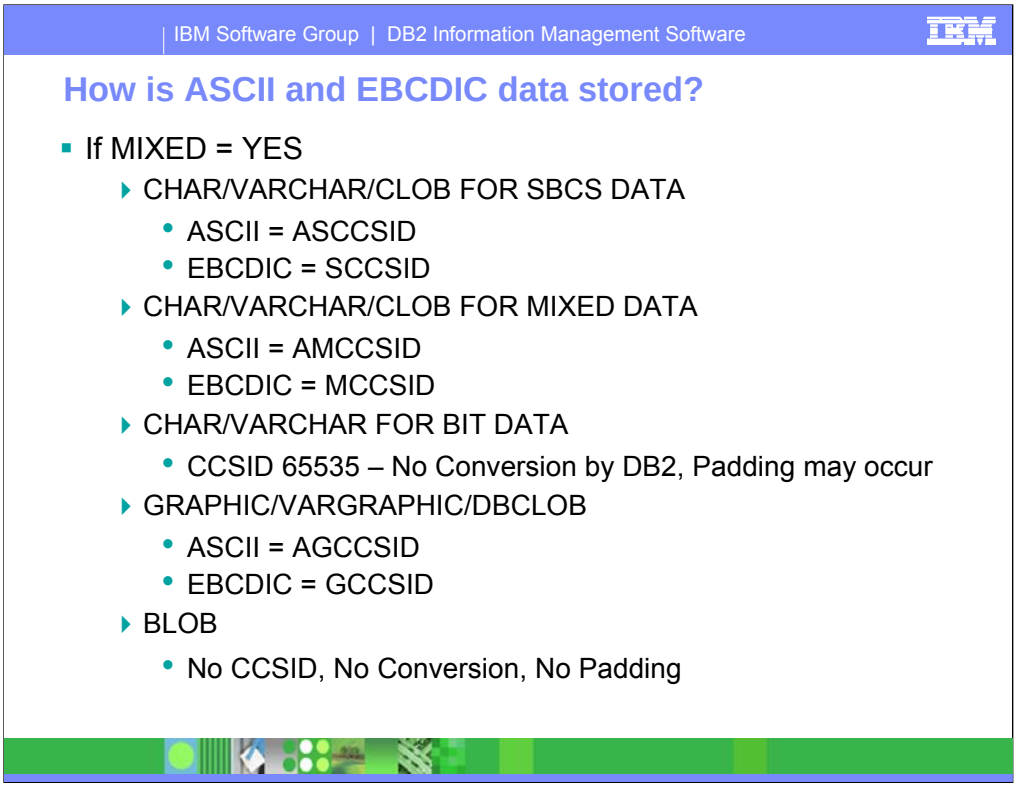
In the latest version of the V8 SQL Reference, this is Table 25. This table is very important because it shows the rules that DB2 uses to determine the result buffer length when a conversion is needed.


IBM
IBM Software Group
How is my data Stored?
How can I tell?
DB2 Information Management Software
ON DEMAND BUSINESS
© 2005 IBM Corporation



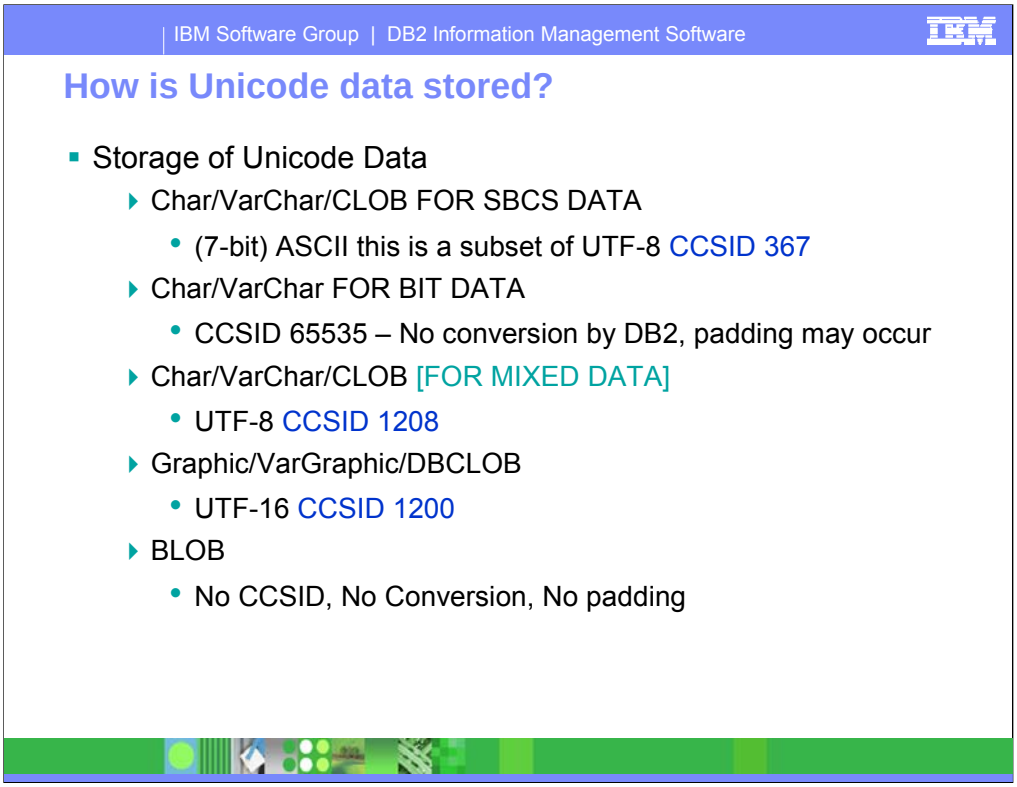
In a MIXED = NO system, only SBCS, CHAR FOR BIT DATA, or BINARY ASCI or EBCDIC data may be stored.

Unicode is not sensitive to the setting of MIXED=YES/NO



In a MIXED = YES system, ASCII or EBCDIC data can be SBCS, BIT, Mixed, Graphic, or BINARY.

Unicode is not sensitive to the setting of MIXED=YES/NO



Data stored in Unicode tables in DB2 will be stored in one of the following CCSIDs
-CCSID 367 is used to store data in columns defined as FOR SBCS DATA.
    -CCISD 367 is a (7 bit ASCII CCSID) that is a subset of UTF-8
-CCSID 65535 data is not subject to conversion, but may be padded
-CCSID 1208 - Unicode UTF-8 - is used by default, or when FOR MIXED DATA is specified
-CCSID 1200 – Unicode UTF-16 – is used when graphic columns are defined in Unicode tables
-BLOB data has no CCSID, is not subject to conversion, and will not be padded by DB2.

How Can I tell?

CCSIDs are stored in the following places

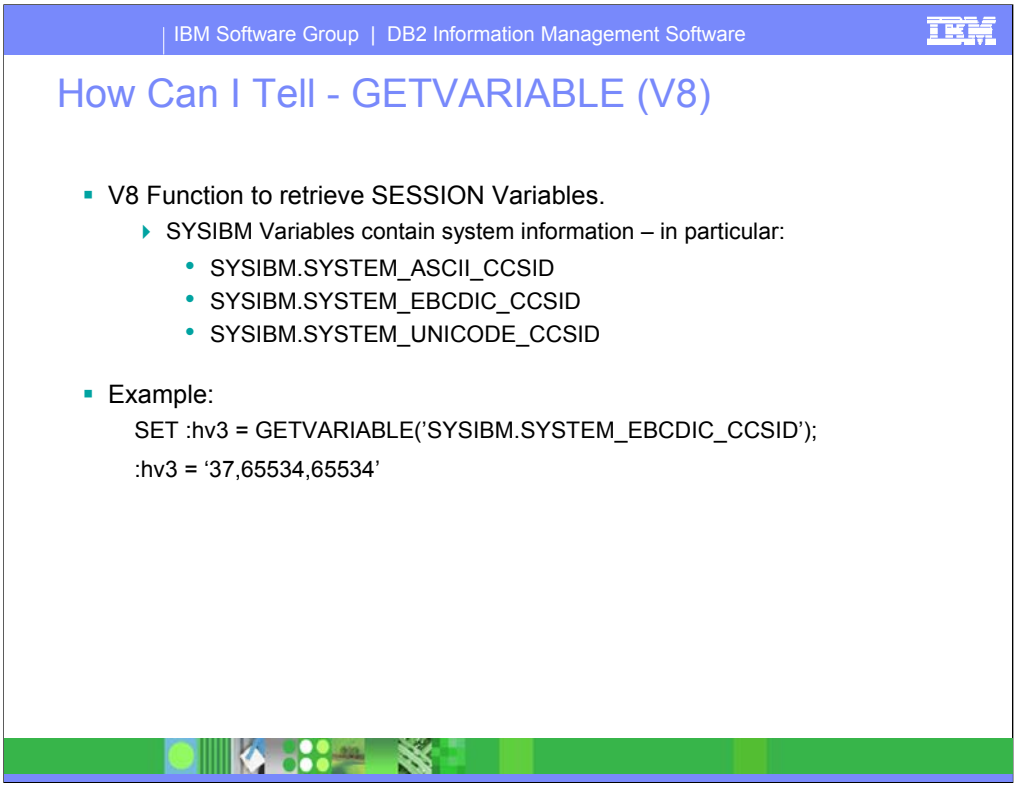
- SYSIBM.SYSDATABASE (V5)
- SYSIBM.SYSCOLUMNS
  - FOREIGNKEY (V2.3) - subtype information
  - CCSID (V8)
- SYSIBM.SYSPACKAGE (V7) – Application Encoding
- SYSIBM.SYSPARMS (V6)
- SYSIBM.SYSPLAN (V7) – Application Encoding
- SYSIMB.SYSROUTINES (V8)
- SYSIBM.SYSTABLES (V8)
- SYSIBM.SYSTABLESPACE (V5)
- SYSIBM.SYSVTREE (V5)
- Plans and Packages (SCT02 and SPT01) – No external
- Directory (DSNDB01) (V5) – No external
- DECP (V2.3)
- BSDS (V8)

In ENCODING_SCHEME column of - Stored as 'A', 'E', 'U', or blank (default)

- SYSIBM.SYSDATATYPES
- SYSIBM.SYSDATABASE
- SYSIBM.SYSPARMS
- SYSIBM.SYSTABLESPACE
- SYSIBM.SYSTABLES

Once we've specified CCSIDs for our system, what does DB2 do with them?
DB2 stores CCSIDs in the Catalog, the directory, in bound statements, the directory, and of course the DECP
The value stored in these areas depends on what release of DB2 was used to create the object, and the value in the DECP at the time the object was created.
If a value is 0, it is assumed that the object is EBCDIC.
In general, DB2 does not support changing of a CCSID once it is specified in a DECP
The exceptions are
Changing from 0 to a valid value

Changing from a CCSID that does not support the EURO symbol to a CCSID that supports the EURO symbol (37 -> 1140 for instance).
Note that this sort of change requires special, disruptive, changes and should be undertaken only after the documentation has been read and the process is thoroughly understood.
Encoding enformation is also stored in some catalog tables



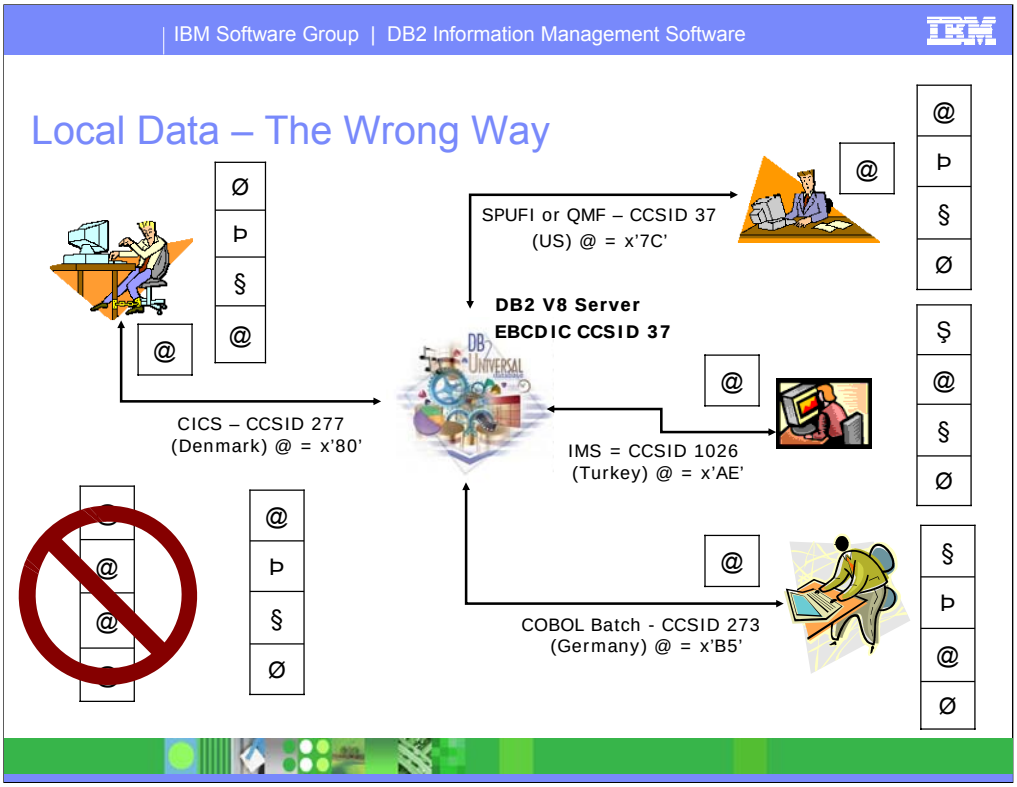
With DB2 V8, the GETVARIABLE BIF can be used to retrieve the CCSIDs that DB2 is using.

How Can I Tell - DSNJU004 – Finding the CCSIDs in the BSDS(V8)

```
//PLM EXEC PGM=DSNJU004
//GROUP DD DSN=DBD1.BSDS01,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  MEMBER *
/*

…
                              SYSTEM CCSIDS
                          18:12:47 MAY 18, 2005
     SYSTEM CCSIDS
     ---------------------
     ASCII SBCS    = 1252
     ASCII MIXED   = 65534
     ASCII DBCS    = 65534
     EBCDIC SBCS   = 37
     EBCDIC MBCS   = 65534
     EBCDIC DBCS   = 65534
     UNICODE SBCS = 367
     UNICODE MBCS = 1208
     UNICODE DBCS = 1200

DSNJ200I  DSNJU004 PRINT LOG UTILITY PROCESSING COMPLETED SUCCESSFULLY
```



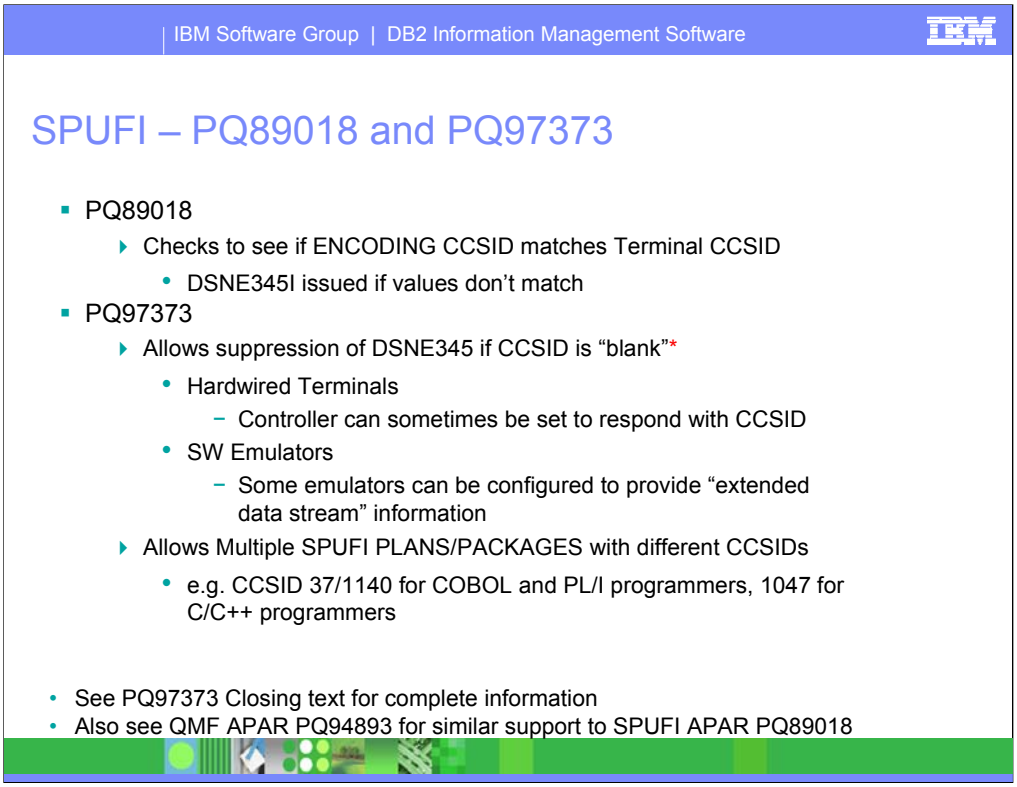
In this example, assume that all the clients are referencing an EBCDIC table (CCSID 37 in this case). Each of the local applications is attempting to insert an "@" into a DB2 table. In general, the results will be incorrect unless DB2 "knows" that the data coming from the CICS, IMS, and COBOL applications is not CCSID 37. There are various ways of the application to tell DB2 the CCSID of the data that they are giving DB2 (in input host variables) or they want to receive from DB2 (in output host variables). These are:

-CCSID overrides in the SQLDA – when using a descriptor

-DECLARE VARIABE – precompiler directive

-ENCODING bind option

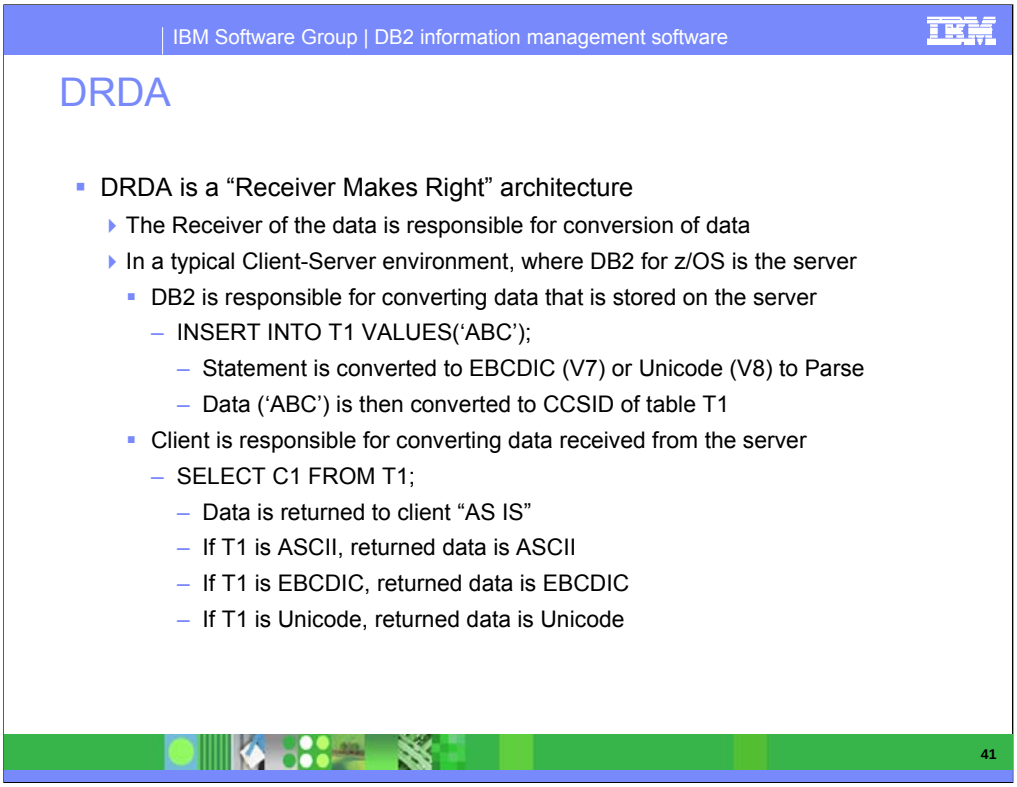
-APPLICATION ENCODNG special register



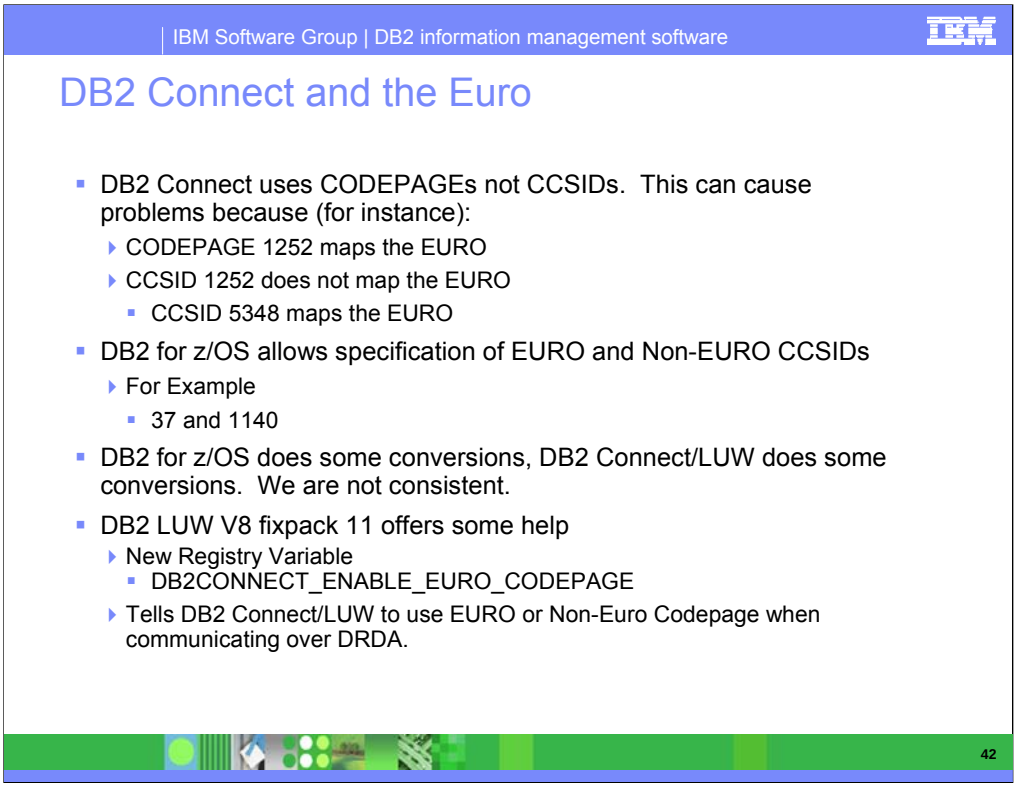
Changes have been made to SPUFI to assist in identifying CCSID issues, and to allow configuration changes so that environments were multiple CCSIDs are used can be accommodated.

Changes similar to PQ89018 have also been made to QMF in PQ94893.

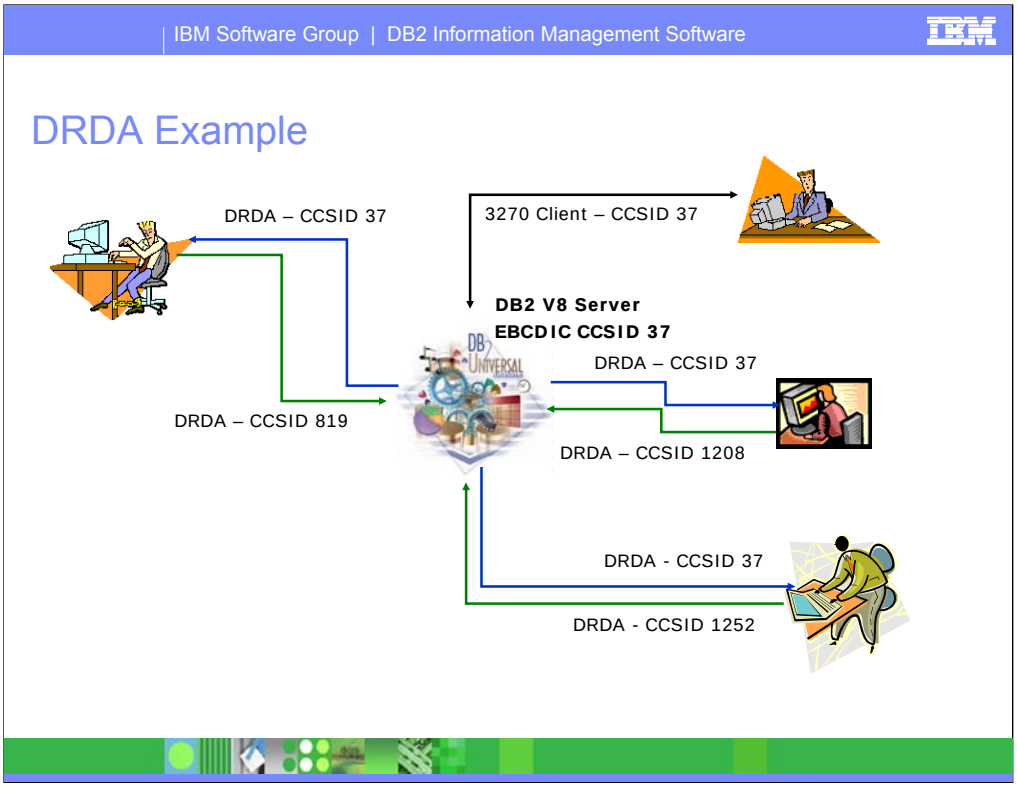

IBM Software Group
DRDA and DB2 Connect
DB2 Information Management Software
ON DEMAND BUSINESS
© 2005 IBM Corporation



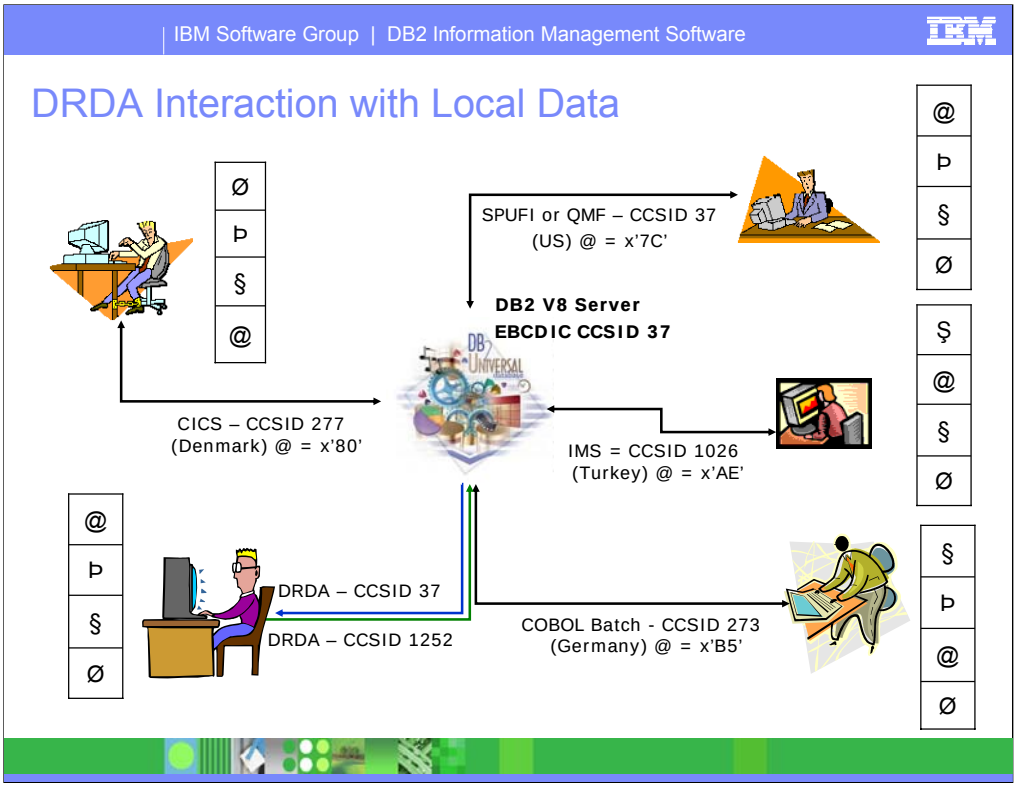
In a DRDA client – server transaction, each side performs ½ of the conversions. The server converts the data it receives from the client to the CCSID that it needs (for instance, the CCSID of the table on an INSERT statement), and the client converts the data it receives to the CCSID it needs (for instance, when a SELECT is issued).



In a DRDA client – server transaction, each side performs ½ of the conversions. The server converts the data it receives from the client to the CCSID that it needs (for instance, the CCSID of the table on an INSERT statement), and the client converts the data it receives to the CCSID it needs (for instance, when a SELECT is issued).

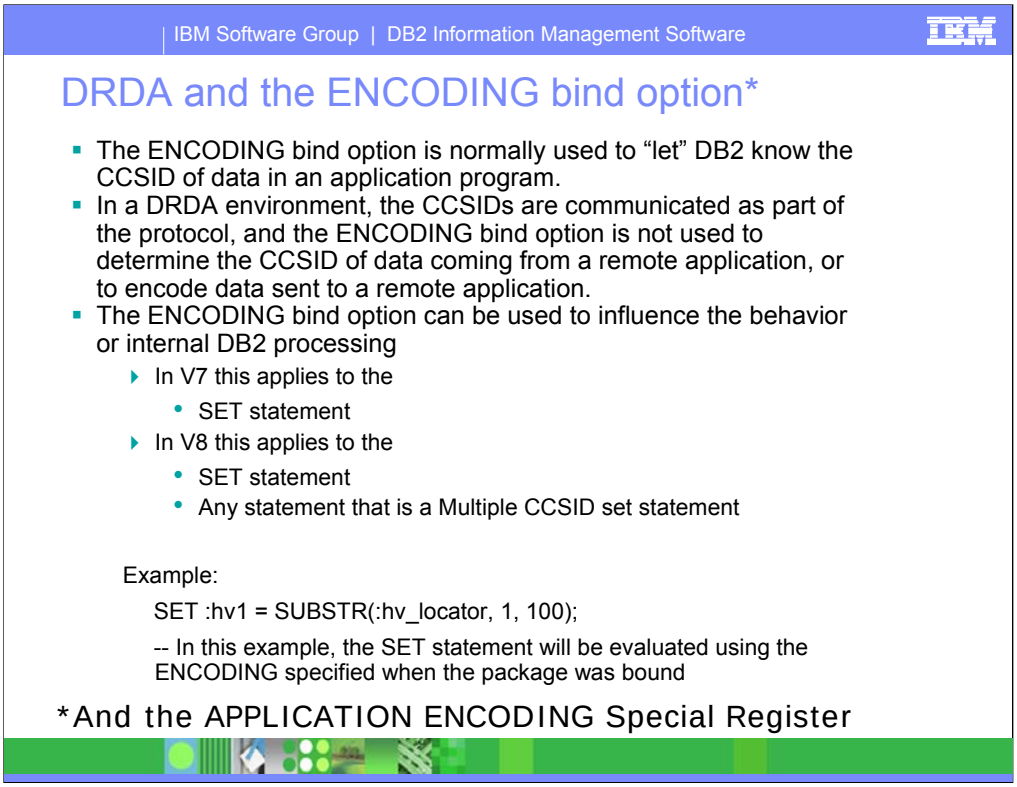


In this example, assume that all the clients are referencing an EBCDIC table (CCSID 37 in this case) the green lines represent data flowing from the client to the server. The server is responsible for converting the data to CCSID 37 to store in the table. The blue lines represent data flows from the server to the clients. The clients are responsible for converting the data from CCSID 37 back to the CCSID they need. The black line represents a local 3270 client – no conversion is necessary in this case.



In this example, assume that all the clients are referencing an EBCDIC table (CCSID 37 in this case). Each of the local applications is attempting to insert an "@" into a DB2 table. In general, the results will be incorrect unless DB2 "knows" that the data coming from the CICS, IMS, and COBOL applications is not CCSID 37.

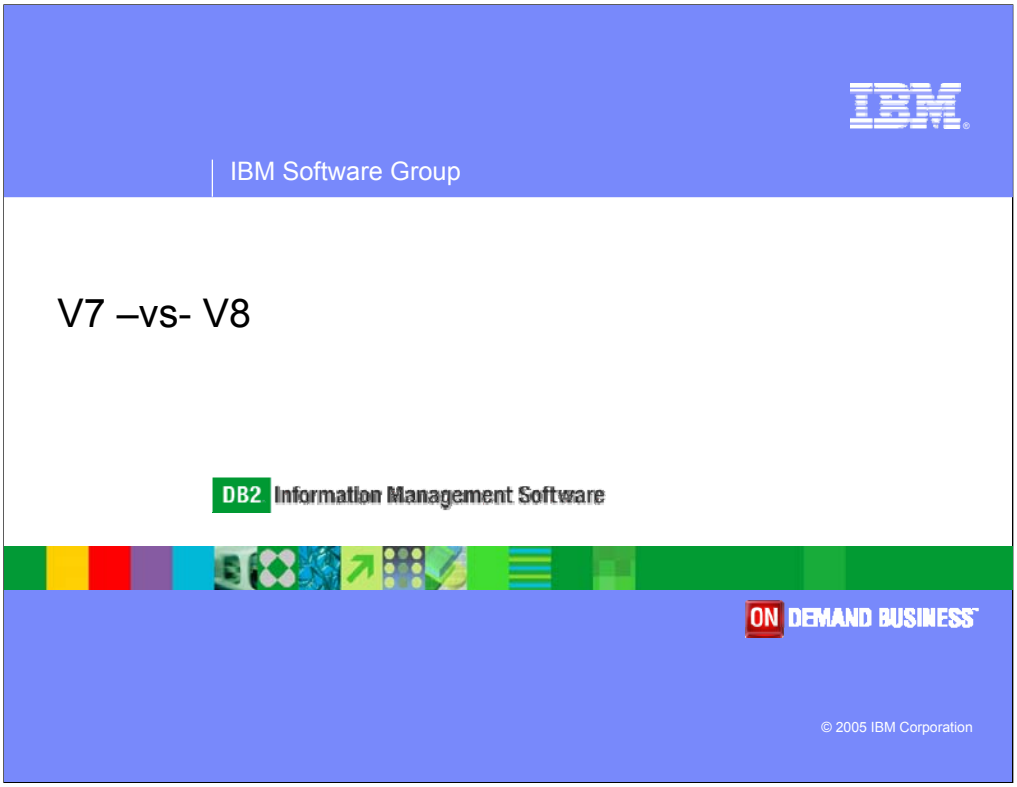
This example demonstrates that remote clients see what a correctly configured local client will see. This is because the EBCDIC data that is sent to the DRDA client is tagged as CCSID 37. When the data is converted from EBCDIC to ASCII CCSID 37 is used as the source of the data. DB2 has no way to tell that some of the data may in fact not be CCSID 37.



The ENCODING bind option is not used in a DRDA application to describe the CCSID of the data. The ENCODING bind option does apply to how DB2 executes certain statements.

In particular the Encoding bind option is used whenever DB2 has a string of variable that has no context (is not used or referenced in a manner that allows DB2 to associate the string or variable with a object that has a known CCSID).

In V7, this applies to the SET statement, and in V8 this applies to the SET statement, and any Multiple CCSID set statemet.


IBM Software Group
V7 –vs- V8
DB2 Information Management Software
ON DEMAND BUSINESS
© 2005 IBM Corporation

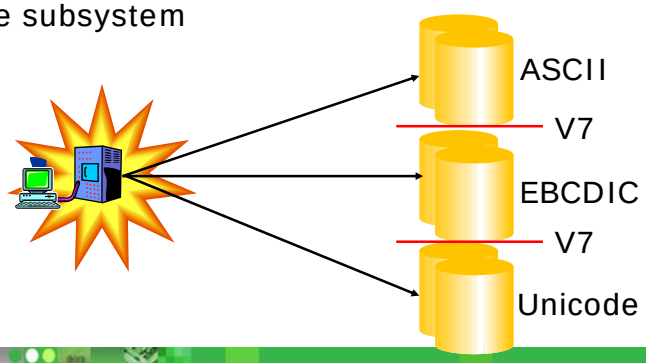
What are CCSIDs used for?

DB2 uses CCSIDs to describe data stored in the DB2 subsystem. CCSIDs are part of the Metadata, such as type and length, that DB2 uses to describe your data

DB2 supports specification of CCSIDs at a subsystem level

Starting with V7, DB2 supports 3 encoding schemes within a single subsystem

- ASCII
- EBCDIC
- UNICODE



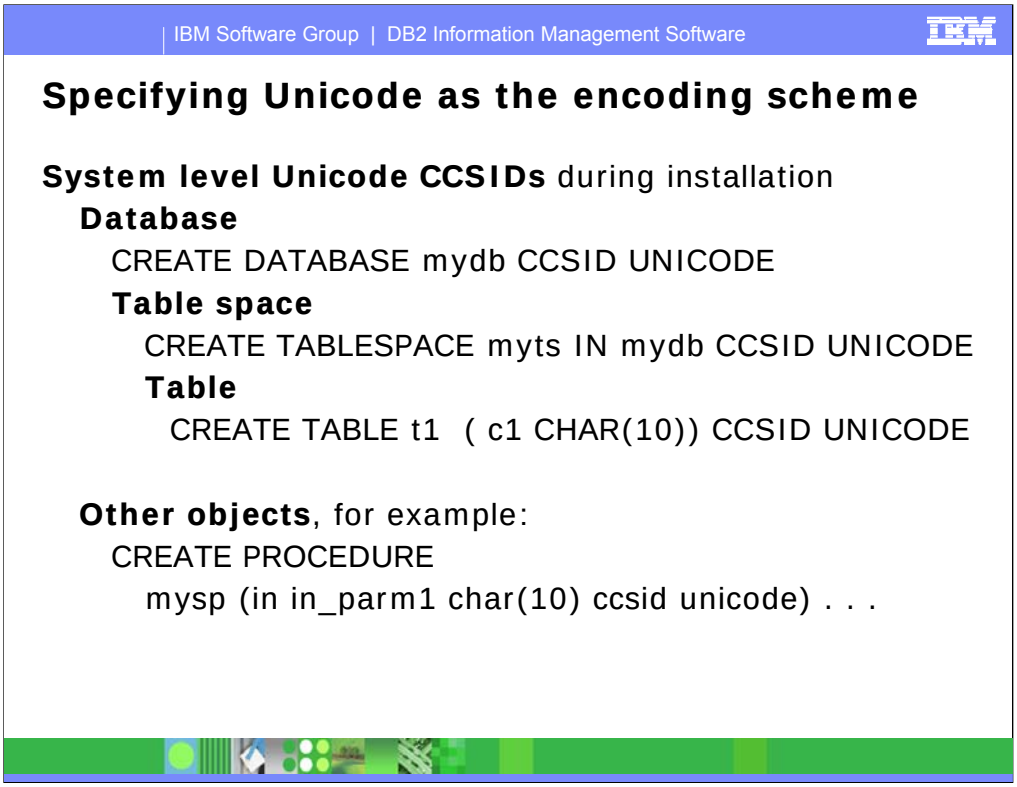
What are CCSIDs used for?

DB2 uses CCSIDs just like we use data type and length. They are part of the metadata that describes the data being stored in DB2.

In DB2 V7 and above, DB2 supports specification of three sets of CCSIDs. These three sets of CCSIDs represent the three encoding schemes (ASCII, EBCDIC, and Unicode), that DB2 supports.

DB2 supports the specification of these CCSIDs at the subsystem level.

Once these values have been specified, they should not be changed.

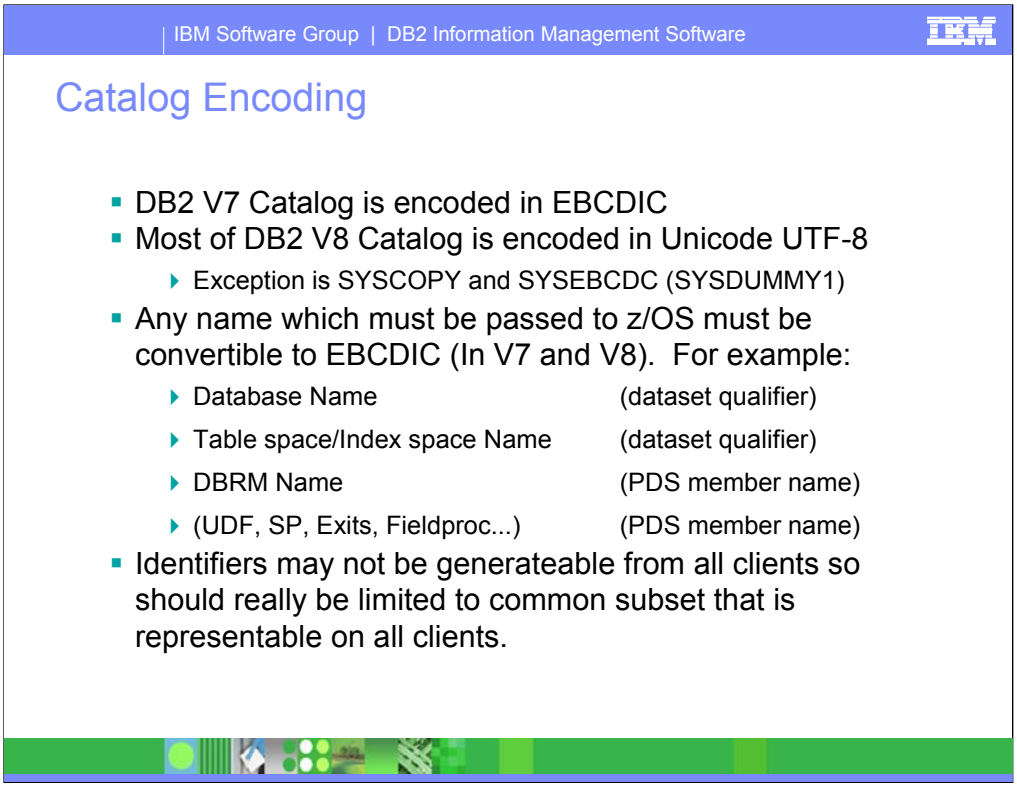


Initial specification is during install on Panel DSNTIPF

Encoding can be specified at the DATABASE, TABLESPACE, or TABLE level. If encoding is not specified, it is inherited.

Other objects such as Procedures, User Defined Functions, and User Defined Types can also specify encoding. This encoding is specified on object creation.

There is no difference between the support for tables in V7 or V8.



The Catalog for DB2 V7 will continue to be in EBCDIC. This means that all statements sent to DB2 will be converted to the system EBCDIC CCSID, and then parsed (for static SQL, this is necessary because statement text is stored in the catalog).

Since statements are converted to EBCDIC, there is a possibility of data loss when the data is converted. Since all DB2 keywords, such as SELECT, are representable on all EBCDIC code pages, there shouldn't be a problem with statement text

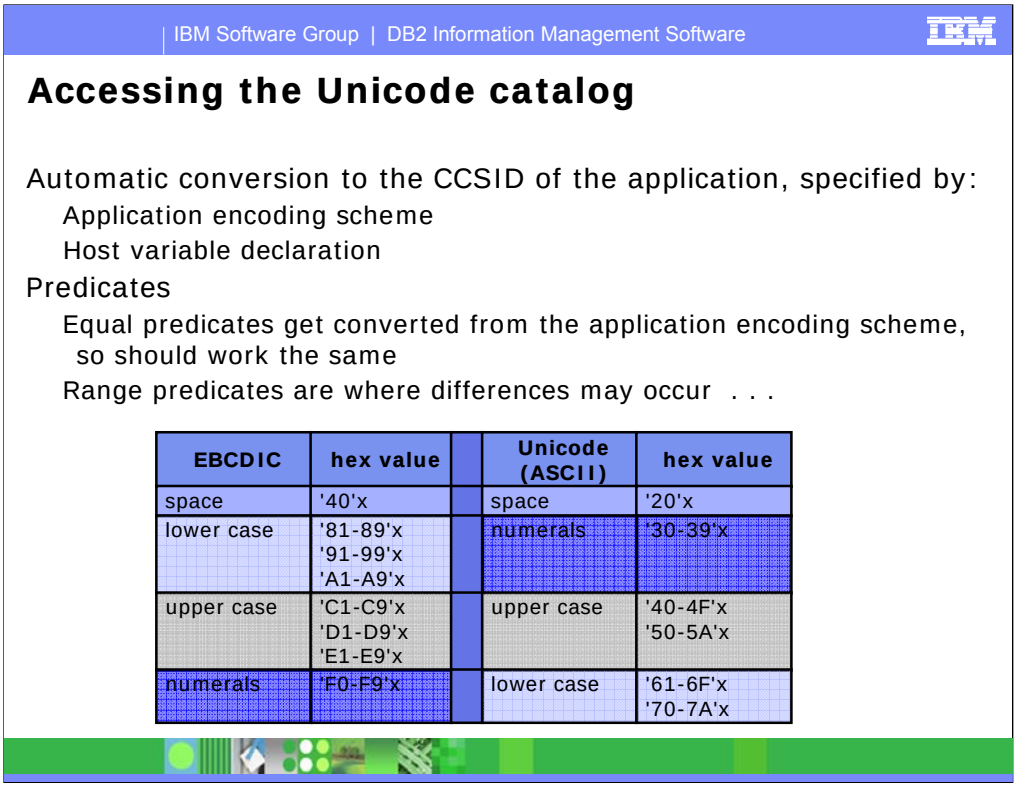
Literals contained in the statement are subject to data loss.

SQLCODE +335 has been added to alert the user of this sort of data loss

The Catalog for DB2 V8 is in Unicode UTF-8.

Since Unicode is a superset of ASCII/EBCDIC, literal values will not be lost on this conversion

The APPLICATION ENCODING CCSID is used to determine the context of literal data.



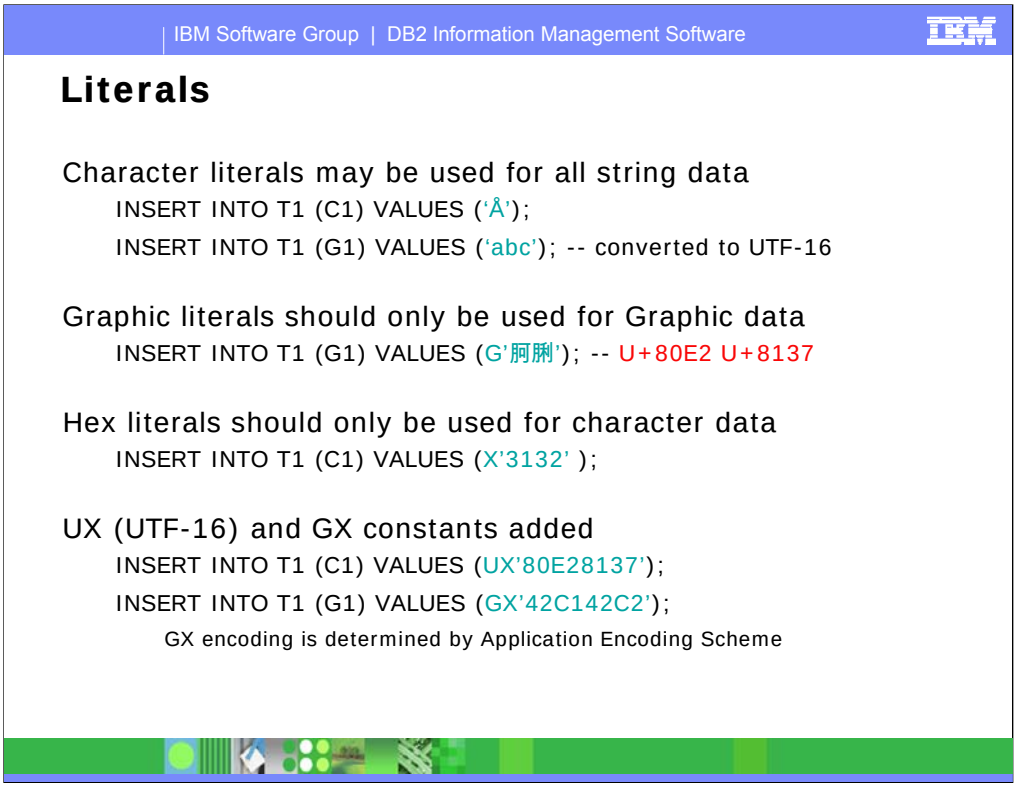
There are many differences you will notice when you access the catalog in DB2 V8 NFM.

Columns may be longer, new columns have been added…

You may not notice the conversion to Unicode – since data will be automatically converted to the your Application CCSID.

However, you may notice differences in range predicates, and ordering.

The DB2 V8 catalog is encoded in Unicode UTF-8, and this may affect some queries.



In DB2 V7, UTF-8 and UTF-16 values may be specified using character literals.

EBCDIC graphic constants “G” constants may also be used.

In DB2 V8, UX and GX constants are added

- UX’ssss’ and GX’ssss’
- In the constant, ssss represents a string from 0 to 32704 hexadecimal digits. The number of characters between the string delimiters must be an even multiple of 4. Each group of 4 digits represents a single UTF-16 (for UX constants) or Graphic (for GX constants) character.

Host Variables and Parameter Markers

Host variable / parameter type:
- ASCII / EBCDIC / Unicode
  - Char
  - Graphic
- Unicode
  - UTF-8
  - UTF-16

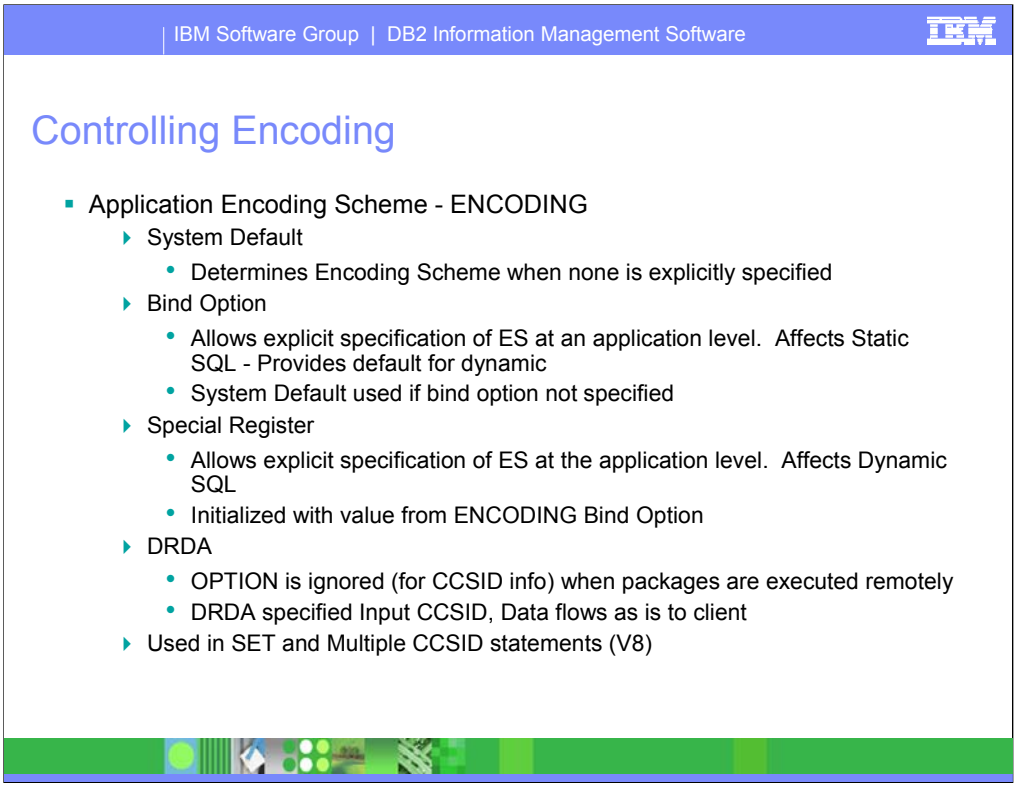
DB2 data storage:
- Unicode
  - UTF-8
  - UTF-16
- ASCII / EBCDIC / Unicode
  - CHAR
  - GRAPHIC

Applications don't need to change just because the back end data store changes

When dealing with Unicode tables, we have torn down the barrier between CHAR and GRAPHIC.

This means your back end data store can be either UTF-8 or UTF-16 and you can use ASCII, EBCDIC, or Unicode character or graphic host variables and DB2 will perform the necessary conversions to/from the CCSID of the host variable even if the host variable doesn't match the column type (for ASCII and EBCDIC back end data stores, in most cases char and graphic are incompatible).

Note: for V7, Graphic Unicode host variables are incompatible with ASCII / EBCDIC SBCS tables. V8 allows this combination.

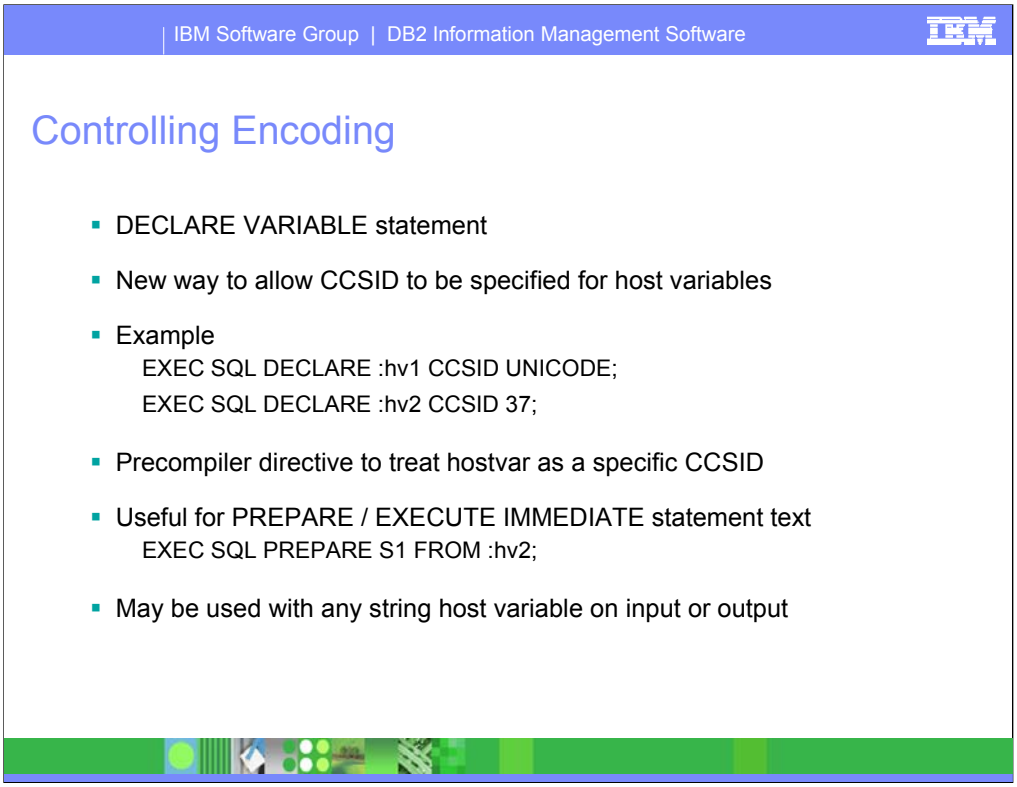


Also new to DB2 V7 is the specification of Application Encoding Scheme.

This allows a default Application Encoding to be specified
Preset to EBCDIC
The Application Encoding Scheme can also be specified on BIND PLAN or PACKAGE
If not specified, the system default value is used for the bind option

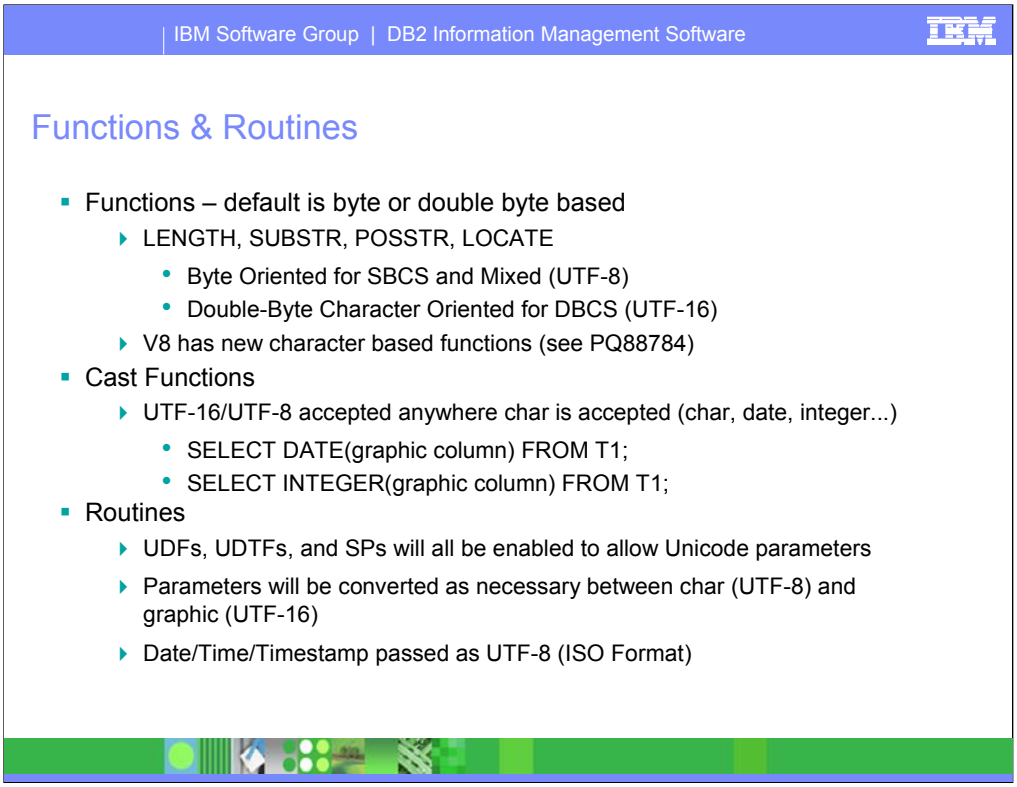
Plans/Packages bound prior to V7 are assumed to be EBCDIC

The option applies to Static SQL
The Application Encoding Scheme special register can be used to affect dynamic SQL
Initial value is the value of the Bind Option.
The SET statement, in V7 and V8, is processed using the ENCODING bind option
In V8, the ENCODING bind option (and special register for dynamic statements) also affects processing of multiple CCSID statements.



The DECLARE VARIABLE statement, added in DB2 V7, can be used to specify the CCSID of a particular host variable.

This is a precompiler directive that causes the precompiler to specify the CCSID of the host variable in any SQLDA that the precompiler generates to reference the host variable. This directive works for both input and output host variables.



All Built In Functions (BIFs) have been extended to support Unicode

Some BIFs, such as LENGTH, SUBSTR, POSSTR, and LOCATE are byte oriented for UTF-8 and Double-Byte character oriented for UTF-16

Many new functions were added in V7, the CCSID_ENCODING function has been added to help users determine the encoding, ASCII, EBCDIC, or UNICODE of a particular CCSID

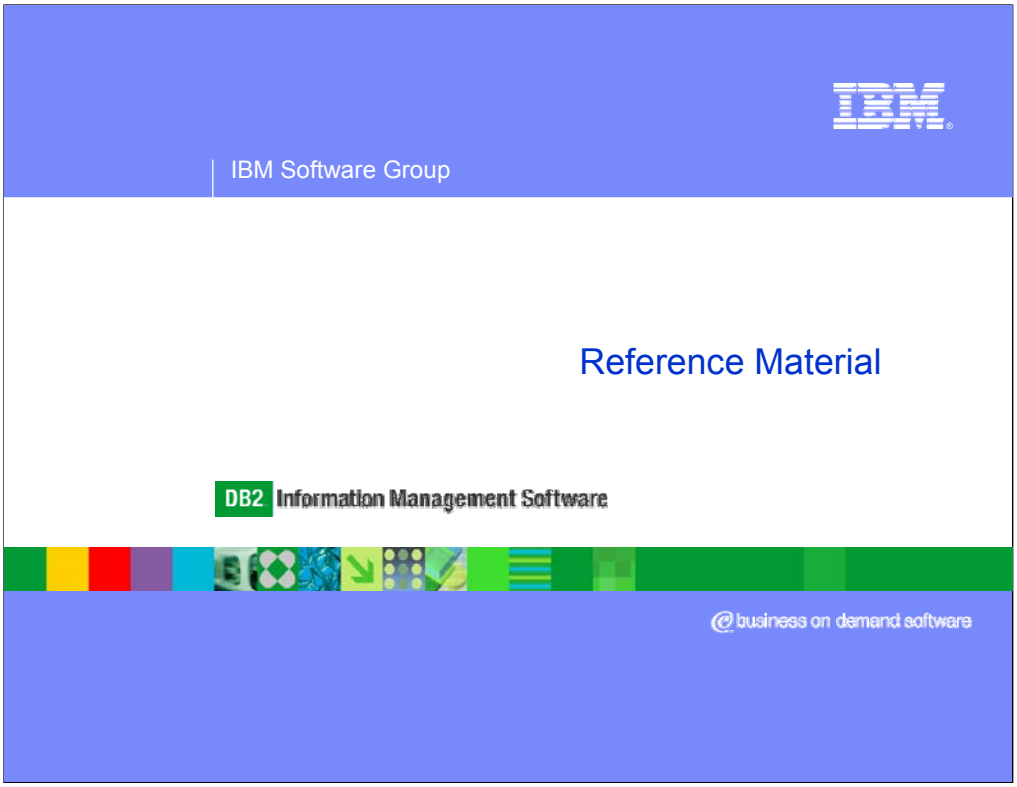
UTF-16 data is accepted in casting type functions such as DATE or INTEGER

Result CCSIDs for functions that return character strings will return UTF-8/CCSID 1208

Session C01

CCSID 101 – What's a CCSID and why do I care?

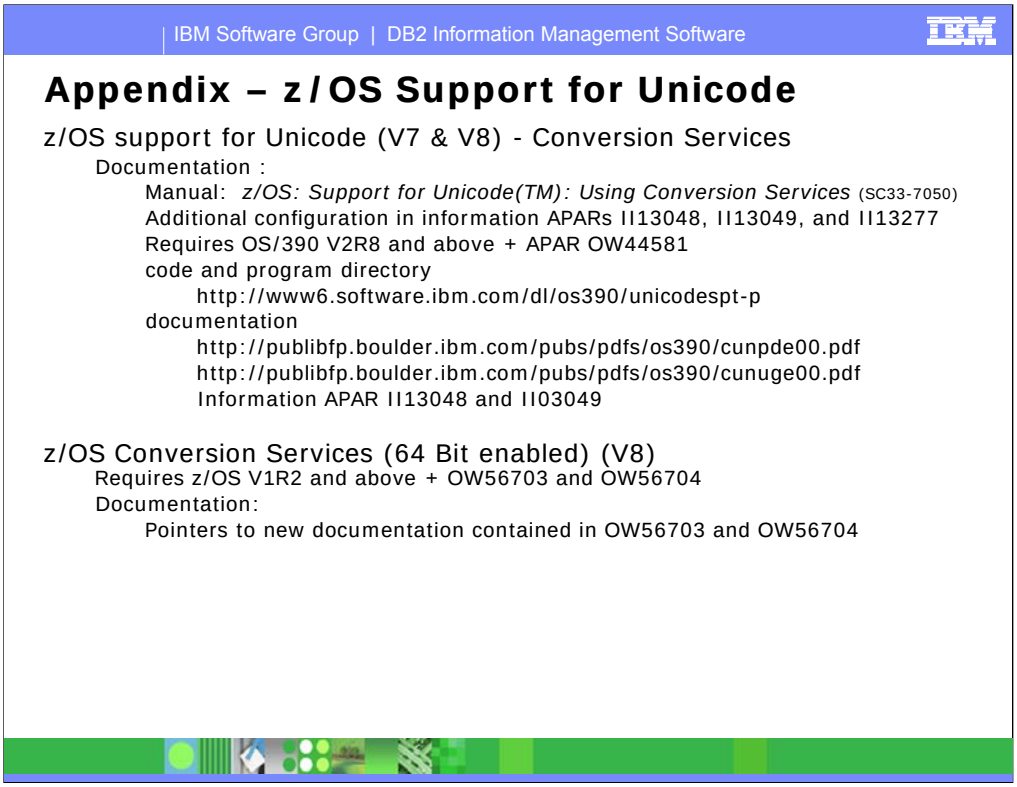
Christopher J. Crone
IBM/DB2 for z/OS Development
cjc@us.ibm.com


IBM Software Group
Reference Material
DB2 Information Management Software
business on demand software

References

DB2 UDB for z/OS Version 8:
Everything You Ever Wanted to Know , ... and More - SG24-6079
DB2 UDB for z/OS Internationalization Guide
http://www.ibm.com/software/data/db2/zos/pdf/ccmstr.pdf
DB2 Universal Database Administration Guide - SC09-2946
Appendix E - National Language Support
The Unicode Standard Version 4.0
The Unicode Consortium - Addison-Wesley - www.unicode.org
Character Data Representation Architecture: Reference & Registry
SC09-2190
National Language Design Guide Volume 2 - SE09-8002
eBusiness Globalization Solution Design Guide, Getting Started
SG24-6851-00

In appendix E of the DB2 UDB Admin Guide, there is a discussion of NLS issues and how to set/override the codepage at the client using the codepage keyword on NT, and LANG variable on AIX.



The z/OS support for Unicode now offers

-Conversion

-Normalization

-Casing

These services are used by DB2 and are also available to Assembler and C/C++ applications.

Appendix - zSeries Unicode Support

The UTF-8 <-> UTF-16 instructions are used when DB2 converts from char <-> graphic. These instructions are used on G5, G6, and zSeries 800,900, 890, and 990:

- CUUTF - Convert UTF-16 to UTF-8
- CUTFU - Convert UTF-8 to UTF-16

The following two instructions are similar to CLCLE and MVCLE. DB2 will use these instructions to perform comparison and padding on UTF-16 data. These instructions are used on zSeries 800, 900, 890, and 990:

- CLCLU - Compare logical long UNICODE
- MVCLU - Move logical long UNICODE

These instructions pack/unpack ASCII (also UNICODE UTF-8) and UNICODE (UTF-16) data. These instructions are used on zSeries 800, 900, 890, and 990:

- PKU - Pack Unicode
- PKA - Pack ASCII
- UNPKU - Unpack Unicode
- UNPKA - Unpack ASCII

These instructions are all used when DB2 performs conversion. DB2 indirectly uses these instructions via the Conversion System Services:

- TRTT - Translate Two to Two
- TRTO - Translate Two to One
- TROT - Translate One to Two
- TROO - Translate One to One

The zSeries 800, 900, 890 and 990 processors provide HW support for Unicode data processing.
The support is detailed here

Conversion Services Example

```
//CUNMIUTL EXEC PGM=CUNMIUTL
//SYSPRINT DD   SYSOUT=*
//TABIN    DD   DISP=SHR,DSN=hlq.SCUNTBL
//SYSIMG   DD   DSN=hlq.IMAGES(CUNIMG00),DISP=SHR
//SYSIN    DD   *
/*** INPUT STATEMENTS FOR THE IMAGE GENERATOR ***/
CONVERSION 0037,1200,ER;  /*EBCDIC  037 -> UTF-16      */
CONVERSION 0037,1208,ER;  /*EBCDIC  037 -> UTF-8       */
CONVERSION 0037,0367,ER;  /*EBCDIC  037 -> ASCII 367   */
CONVERSION 1200,0037,ER;  /*UTF-16      -> EBCDIC  037 */
CONVERSION 1208,0037,ER;  /*UTF-8       -> EBCDIC  037 */
CONVERSION 0367,0037,ER;  /*ASCII 367   -> EBCDIC  037 */
CONVERSION 1252,1200,ER;  /*ASCII  1252 -> UTF-16      */
CONVERSION 1252,1208,ER;  /*ASCII  1252 -> UTF-8       */
CONVERSION 1252,0367,ER;  /*ASCII  1252 -> ASCII 367   */
CONVERSION 1200,1252,ER;  /*UTF-16      -> ASCII  1252 */
CONVERSION 1208,1252,ER;  /*UTF-8       -> ASCII  1252 */
CONVERSION 0367,1252,ER;  /*ASCII 367   -> ASCII  1252 */
CONVERSION 1208,1200,ER;  /*UTF-8       -> UTF-16      */
CONVERSION 0367,1200,ER;  /*ASCII 367   -> UTF-16      */
CONVERSION 1200,1208,ER;  /*UTF-16       -> UTF-8      */
CONVERSION 0367,1208,ER;  /*ASCII 367    -> UTF-8      */
CONVERSION 1200,0367,ER;  /*UTF-16       -> ASCII 367  */
CONVERSION 1208,0367,ER;  /*UTF-8        -> ASCII 367  */
/*
```

Here's an example of the CUNJIUTL utility

Note the specification of ER (enforced subset, round trip) after each CCSID pair. The ER specification is required by DB2.

See II13048 and II13049 for further information on setting up the z/OS conversion services with DB2.

Conversion Services Example Display

```
14.34.14        d uni,all
14.34.15        CUN3000I 14.34.14 UNI DISPLAY 097
   ENVIRONMENT: CREATED       12/11/2002 AT 09.13.53
                MODIFIED      12/11/2002 AT 09.13.53
                IMAGE CREATED 12/06/2002 AT 17.10.01
       SERVICE: CUNMCNV   CUNMCASE
       STORAGE: ACTIVE       50 PAGES
                LIMIT    524287 PAGES
      CASECONV: NONE
    CONVERSION: 00037-00367-ER              00037-01208-ER
                00037-01200(13488)-ER       00367-00037-ER
                00367-01208-ER              00367-01200(13488)-ER
                00367-01252-ER              01200(13488)-00037-ER
                01200(13488)-00367-ER       01200-01208-ER
                01200(13488)-01252-ER       01208-00037-ER
                01208-00367-ER              01208-01200-ER
                01208-01252-ER              01252-00367-ER
                01252-01200(13488)-ER       01252-01208-ER
```

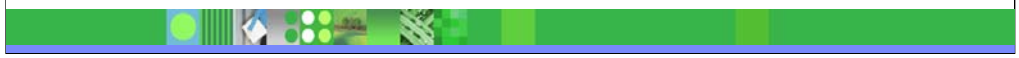
Here's an example of output from a display UNI command.
Note when the image was created is displayed

Also, the number of pages used by the image is also displayed. This is important because these pages are page fixed so they are taking up dedicated memory space on the machine

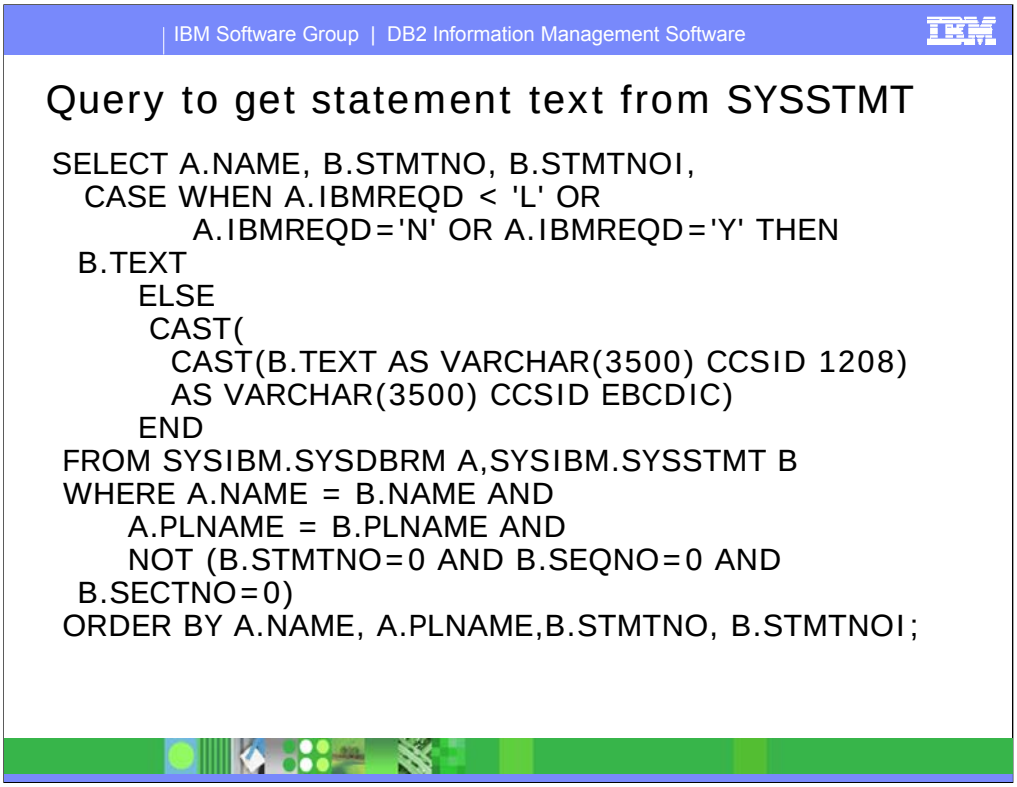
Finally, a list of conversions that are supported in this image are displayed.
Note for CCSID 1200 conversions the base CCSID that the conversion was created from is also displayed in parenthesis.

SYSSTMT and SYSPACKSTMT

Statement text in SYSSTMT and SYSPACKSTMT
- EBCDIC for
  - Applications Precompiled Prior To DB2 V8
  - Applications Precompiled in NEWFUN(NO) mode
- Unicode for
  - Applications Precompiled in NEWFUN(YES) mode

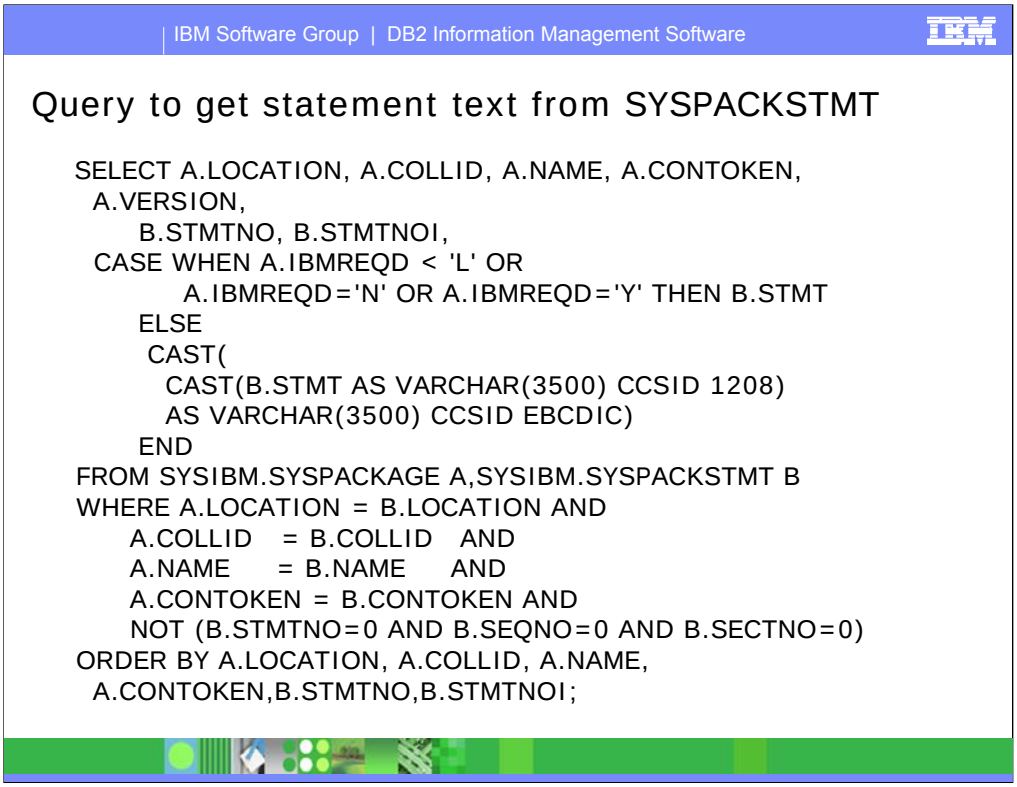


The statement text in SYSSTMT and SYSPACKSTMT is essentially Binary.
DB2 knows how to access the information, but applications will have trouble accessing these columns because the encoding changes depending on the NEWFUN option that was used to precompile the application.



An SQL statement similar to the above may be used to get SQL statement text from SYSSTMT.

Please note that due to the nature of the data (some of the data is not textual data) in the TEXT column, warnings and errors are possible.



An SQL statement similar to the above may be used to get SQL statement text from SYSPACKSTMT.

Please note that due to the nature of the data (some of the data is not textual data) in the STMT column, warnings and errors are possible.