X/Open System Verification Suite

Pseudo-language Specification

VSXgen1.3 May 1998

The Open Group

Any comments relating to the material in this document may be sent by electronic mail to the VSXgen support team at:

`VSX_support@rdg.opengroup.org`

1. INTRODUCTION

This document describes the test locales, or *pseudo-languages*, used by VSXgen-based test suites to verify interfaces which have localisation dependencies. As well as giving details of the required contents of each pseudo-language, it explains the reasoning behind the usage of pseudo-languages.

2. LOCALISATION TESTING OVERVIEW

2.1 LOCALE DEFINITIONS

The Single UNIX Specification and its predecessors do not provide definitions for the internal format of locale information such as character classifications and collating sequences. They only define the file format for the `localedef` command which creates a new locale in the language database available to an application. The manner in which this data is stored is implementation specific. The specifications are otherwise limited to the effect of the current language setting on the appropriate functions and the way in which an application may alter the current language setting.

The language database supplied with a particular implementation may contain an extensive set of languages or, alternatively, may only contain a minimal set of languages. The specifications do not demand that certain languages or dialects are present on a conforming system and VSX cannot assume that any given set of languages will be present.

This clearly raises a problem for the verification suite; to verify localisation aspects thoroughly, at least one instance of the various capabilities is needed. For example, there must be instances of 2–to–1 and 1–to–2 character mappings in the collating sequence tables to verify that the implementation supports these capabilities. Since the set of languages provided by an implementation is not mandated, it is not possible to guarantee that all of these situations can be generated by using data from a specific implementation's language database.

To overcome this problem, the concept of a pseudo-language has been introduced for the purposes of verification. The installation of the VSX pseudo-languages is a prerequisite to the successful execution of the verification suite.

2.2 THE VERIFICATION PSEUDO-LANGUAGES

The pseudo-languages are designed to cater for the testing of 8-bit transparency in the functions which make use of the language database. They are also designed to allow thorough testing of all the capabilities defined in the specifications; this cannot be achieved by using only one test language. It is important to test that when a change of language is successfully requested by `setlocale()`, the language change is correctly reflected when other functions are called. For this reason five languages must be installed in the language database of the implementation being tested. These pseudo-languages contain all the necessary elements to allow the verification of localisation features.

Each pseudo-language includes the characters of the portable character set with their normal encodings (i.e. the same encodings that they have in the C locale). This necessitates the encodings for the additional international characters to be configurable, so that they do not conflict with the characters of the portable character set or the control character set, given arbitrary encodings for these two sets.

Two different encoding variations are supplied with VSX. The first, which is the default, is suitable for systems on which the characters of the portable character set and control character set all lie in the range 0x0–0x7f (as in ASCII). The second is suitable for systems which use EBCDIC. If neither of these is appropriate for your system, you

must design your own encodings.

2.3 PSEUDO-LANGUAGE INSTALLATION

VSX expects you to install the pseudo-languages into the language database used on your implementation. If your system supports the `localedef` utility, and uses either the ASCII or EBCDIC character set, you should be able to do this by making use of the `localedef` sources supplied with VSX, with little or no modification. If your system does not support `localedef`, you may still find these files useful as a starting point for producing the pseudo-language definitions in a format suitable for your system.

VSX is implemented so that the absence of the pseudo-languages does not effect the installation or building procedures of the verification suite and does not prohibit the execution of the verification suite. Where a particular test depends on the correct installation of a pseudo-language, this test gives an `unresolved` or `uninitiated` result during the execution of the suite if the pseudo-language is either unavailable or incorrectly installed. You can re-run the affected tests, without rebuilding, after the pseudo-languages have been correctly installed. Note however that if you change the character encodings configured in the files `SRC/INC/pslcodes.h` and `SRC/INC/ctrlcodes.h` then you must repeat the VSX installation and building stages in order for the new encodings to be used.

2.4 SPECIAL CONSIDERATIONS

Since the pseudo-languages contain additional alphanumeric characters that are not printable in the C locale, a problem arises in the printing of error messages when a test fails. It is not desirable for the journal to contain non-printing characters and so the output must be made in a different form.

This problem is resolved by translating the affected characters from the pseudo-language into a printable form whenever possible. The translation represents pseudo-language characters in the following way:

- Accented characters are printed as a character pair, with the alphabetic character followed by an accent represented as follows:
  - — Umlaut is represented by `"`.
  - — Acute accent is represented by `'`.
  - — Grave accent is represented by `‘`.
  - — Cedilla is represented by `~`.
  - — Circumflex is represented by `^`.
- The eszet character (β) is represented by `&`.
- Control characters are printed in C string literal form (\b, \n, etc.) where appropriate.
- The backslash character (\) is represented by `\\`.
- Other characters are printed in the form `\`*nnn* where *nnn* is an octal number.

This will enable you to compare expected and actual values. It will, however, be necessary on occasions to refer to the pseudo-language definitions, below, to decipher the output.

3. PSEUDO-LANGUAGE DEFINITION

3.1 INTRODUCTION

VSX uses five pseudo-languages. The first four of these contain enough differences from each other to ensure that the current locale setting is correctly referenced. The fifth language is, effectively, a null language definition, and tests the behaviour of interfaces when the required language data is not available for the current locale setting.

The locale names for the pseudo-languages are:

Pseudo-Language Number	Locale Name
1	`VSX4L1`
2	`VSX4L2`
3	`VSX4L3`
4	`VSX4L3@dict`
5	`VSX4L0`

3.2 CHARACTER ENCODINGS FOR PSEUDO-LANGUAGES

Each pseudo-language contains both the portable character set and the control character set with their normal encodings, plus a number of additional characters. The character encodings for the additional characters are specified in the file `SRC/INC/pslcodes.h`, which can be altered by the user if the default encodings conflict with the encodings for the portable character set or the control character set. The hexadecimal values are shown below for the default encodings, which are suitable for systems where the characters of the portable character set and control character set have values in the range 0x0 to 0x7f (as in the ASCII character set).

3.2.1 Default encodings for `VSX4L1` and `VSX4L2`

	8	9	A	B	C	D	E	F
C		Ä				ä		
D		Ö				ö		
E		Ü				ü		
F		cedilla				β		

3.2.2 Default encodings for `VSX4L3` and `VSX4L3@dict`

	8	9	A	B	C	D	E	F
0					À		à	
1					Á		á	
2					Â	Ò	â	ò
3					Ã	Ó	ã	ó
4					Ä	Ô	ä	ô
5					Å	Õ	å	õ
6						Ö		ö
7					Ç		ç	
8					È		è	
9					É	Ù	é	ù
A					Ê	Ú	ê	ú
B					Ë	Û	ë	û
C					Ì	Ü	î	ü
D					Í	Ý	í	ý
E					Î		î	
F					Ï	β	ï	

3.2.3 Pseudo-Language 5 (`VSX4L0`)

The pseudo-language `VSX4L0` has no additional characters. It should be installed with the same character encodings as the C locale.

3.3 CHARACTER CLASSIFICATIONS FOR PSEUDO-LANGUAGES

In all of the pseudo-languages the characters of the portable character set have the same classifications and uppercase/lowercase pairings as in the C locale, and the characters of the control character set all have the classification `iscntrl`. The classifications for the additional characters in each pseudo-language are given below. The uppercase/lowercase pairings are defined such that each uppercase accented character is paired with the lowercase character formed from the equivalent lowercase base character and the same accent. For example (Ä–ä), (É–é), and so on. The lowercase character β does not have an uppercase equivalent.

Characters in the hex range 0x0 to 0x7F have the classification `isascii` (not needed for POSIX modes).

3.3.1 Pseudo-Languages 1 and 2 (`VSX4L1` and `VSX4L2`)

Characters	**Classification**
Ä,Ö,Ü	isalpha, isupper, isalnum, isprint, isgraph
ä,ö,ü,β	isalpha, islower, isalnum, isprint, isgraph
cedilla	ispunct, isprint, isgraph

3.3.2 Pseudo-Languages 3 and 4 (`VSX4L3` and `VSX4L3@dict`)

Characters	**Classification**
Ä,Á,Â,À,Ã,Å,Ç	isalpha, isupper, isalnum, isprint, isgraph
Ë,É,Ê,È,Ï,Í,Ì	isalpha, isupper, isalnum, isprint, isgraph
Ö,Ó,Ô,Ò,Õ	isalpha, isupper, isalnum, isprint, isgraph
Ü,Ú,Û,Ù,Ý	isalpha, isupper, isalnum, isprint, isgraph
ä,á,â,à,ã,å,ç	isalpha, islower, isalnum, isprint, isgraph
ë,é,ê,è,ï,ï,î,î	isalpha, islower, isalnum, isprint, isgraph
ö,ó,ô,ò,õ	isalpha, islower, isalnum, isprint, isgraph
ü,ú,û,ù,ý,β	isalpha, islower, isalnum, isprint, isgraph

3.4 LANGUAGE INFORMATION CONSTANTS FOR PSEUDO-LANGUAGES

Language Constant	Languages 1,3 VSX4L1 VSX4L3	Languages 2,4 VSX4L2 VSX4L3@dict
D_T_FMT	%I.%M.%S %p %m/%d/%y	%d %m %Y %H:%M:%S
D_FMT	%m/%d/%y	%d %m %Y
T_FMT	%I.%M.%S %p	%H:%M:%S
T_FMT_AMPM	%I.%M.%S %p	%I:%M:%S %p
AM_STR	VM	am
PM_STR	NM	pm
DAY-1	Sonntag	dimanche
DAY-2	Montag	lundi
DAY-3	Dienstag	mardi
DAY-4	Mittwoch	mercredi
DAY-5	Donnerstag	jeudi
DAY-6	Freitag	vendredi
DAY-7	Samstag	samedi
ABDAY-1	Son	dim
ABDAY-2	Mon	lun
ABDAY-3	Die	mar
ABDAY-4	Mit	mer
ABDAY-5	Don	jeu
ABDAY-6	Fre	ven
ABDAY-7	Sam	sam
MON-1	Januar	janvier
MON-2	Februar	février
MON-3	März	mars
MON-4	April	avril
MON-5	Mai	mai
MON-6	Juni	juin
MON-7	Juli	juillet
MON-8	August	août
MON-9	September	septembre
MON-10	Oktober	octobre
MON-11	November	novembre
MON-12	Dezember	décembre
ABMON-1	Jan	jan
ABMON-2	Feb	fév
ABMON-3	März	mars
ABMON-4	Apr	avr
ABMON-5	Mai	mai
ABMON-6	Juni	juin
ABMON-7	Juli	juil
ABMON-8	Aug	août
ABMON-9	Sept	sept
ABMON-10	Okt	oct
ABMON-11	Nov	nov
ABMON-12	Dez	déc
RADIXCHAR	,	I
THOUSEP	.	,
YESSTR	j	o
NOSTR	n	n
CRNCYSTR	-Dm	+Fr

Note that THOUSEP, YESSTR, NOSTR and CRNCYSTR are not used in POSIX and FIPS test modes, and T_FMT_AMPM is not used in POSIX, FIPS and XPG3 test modes.

Characters containing acute and circumflex accents are a true single character in language 4 (VSX4L3@dict), but in language 2 (VSX4L2) they are constructed as a two-character sequence consisting of the base alphabetic character followed by the accent.

All of the above information strings should either be made unavailable or be set to the empty string for pseudo-language 5 (VSX4L0), with the exception that if your system does not allow RADIXCHAR to be unavailable or to be an empty string, then it can be set to ‘‘.’’.

The following items are only required for XPG4 and UNIX modes. The symbol □ is used to represent a space character.

Language Constant	Languages 1,3 VSX4L1 VSX4L3	Languages 2,4 VSX4L2 VSX4L3@dict
decimal_point	","	"I"
thousands_sep	"."	","
grouping	""	"\1\2\3\4"
int_curr_symbol	"DDM□"	"FRF□"
currency_symbol	"Dm"	"Fr"
mon_decimal_point	"^"	"*"
mon_thousands_sep	"□"	"-"
mon_grouping	"\3"	"\4"
positive_sign	""	"+"
negative_sign	"C"	"-"
int_frac_digits	2	1
frac_digits	3	2
p_cs_precedes	1	0
p_sep_by_space	0	0
n_cs_precedes	1	0
n_sep_by_space	0	1
p_sign_posn	3	4
n_sign_posn	3	2

For pseudo-language 5 (VSX4L0)), all items except decimal_point must be made unavailable, so that localeconv() will return the structure members with string values as an empty string, and the members with numeric values as CHAR_MAX. If your system allows it, you should either make decimal_point unavailable or specify its value as an empty string. If not, then decimal_point should be set to ".".

3.5 COLLATION SEQUENCE TABLES

The collating sequence information uses a primary and secondary collating sequence system. The primary collating sequence ignores accents and case priority. Where two strings collate equally according to their primary sequence, the secondary sequence is used to determine order.

3.5.1 Pseudo-Language 1 (VSX4L1)

Primary Element	Secondary Elements (in order)	Notes
-		- is considered as a don't care character
a	ä,a,à,â,Ä,A,À,Â	
b	b,B	
c	c,ç,C,Ç	
ch		ch collates after c as a 2-1 mapping
d	d,D	
e	e,é,è,ê,E,É,È,Ê	
f	f,F	
g	g,G	
h	h,H	
i	i,î,I,Î	
j	j,J	
k	k,K	
l	l,L	
m	m,M	
n	n,N	
o	ö,o,ô,Ö,O,Ô	
p	p,P	
q	q,Q	
r	r,R	
s	s,S	
β		β collates equally with ss as a 1-2 mapping
t	t,T	
u	ü,u,Ü,U	
v	v,V	
w	w,W	
x	x,X	
y	y,Y	
z	z,Z	
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
,	,	
+	+	

Primary Element	Secondary Elements (in order)	Notes
(	(	
)	)	
*	*	
%	%	
#	#	
$	$	
space	space	

Characters containing acute, grave, cedilla and circumflex accents are constructed as a two-character sequence consisting of the base alphabetic character followed by the accent.

3.5.2 Pseudo-Language 2 (VSX4L2)

Primary Element	Secondary Elements (in order)	Notes
-		- is considered as a don't care character
a	ä,a,à,â	
b	b	
c	c,ç	
ch		ch collates after c as a 2-1 mapping
d	d	
e	e,é,è,ê	
f	f	
g	g	
h	h	
i	i,î	
j	j	
k	k	
l	l	
m	m	
n	n	
o	ö,o,ô	
p	p	
q	q	
r	r	
s	s	
β		β collates equally with ss as a 1-2 mapping
t	t	
u	ü,u	
v	v	
w	w	
x	x	
y	y	
z	z	
A	Ä,A,À,Â	
B	B	
C	C,Ç	
Ch		Ch collates after C as a 2-1 mapping
D	D	
E	E,É,È,Ê	
F	F	
G	G	
H	H	
I	I,Î	
J	J	
K	K	
L	L	
M	M	
N	N	
O	Ö,O,Ô	
P	P	

Primary Element	Secondary Elements (in order)	Notes
Q	Q	
R	R	
S	S	
T	T	
U	Ü,U	
V	V	
W	W	
X	X	
Y	Y	
Z	Z	
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
,	,	
+	+	
(	(	
)	)	
*	*	
%	%	
#	#	
$	$	
space	space	

Characters containing acute, grave, cedilla and circumflex accents are constructed as a two-character sequence consisting of the base alphabetic character followed by the accent.

3.5.3 Pseudo-Language 3 (VSX4L3)

Primary Element	Secondary Elements (in order)	Notes
-		- is considered as a don't care character
a	a,à,á,â,ã,ä,å	
b	b	
c	c,ç	
ch		ch collates after c as a 2-1 mapping
d	d	
e	e,è,é,ê,ë	
f	f	
g	g	
h	h	
i	i,î,ï,í,î,ï	
j	j	
k	k	
l	l	
m	m	
n	n	
o	o,ò,ó,ô,õ,ö	
p	p	
q	q	
r	r	
s	s	
β		β collates equally with ss as a 1-2 mapping
t	t	
u	u,ù,ú,û,ü	
v	v	
w	w	
x	x	
y	y,ý	
z	z	
A	A,À,Á,Â,Ã,Ä,Å	
B	B	
C	C,Ç	
D	D	
E	E,È,É,Ê,Ë	
F	F	
G	G	
H	H	
I	I,Ì,Í,Î,Ï	
J	J	
K	K	
L	L	
M	M	
N	N	
O	O,Ò,Ó,Ô,Õ,Ö	
P	P	
Q	Q	
R	R	

Primary Element	Secondary Elements (in order)	Notes
S	S	
T	T	
U	U,Ù,Ú,Û,Ü	
V	V	
W	W	
X	X	
Y	Y,Ý	
Z	Z	

3.5.4 Pseudo-Language 4 (VSX4L3@dict)

Primary Element	Secondary Elements (in order)	Notes
-		- is considered as a don't care character
a	A,À,Á,Â,Ã,Ä,Å, a,à,á,â,ã,ä,å	
b	B,b	
c	C,Ç,c,ç	
ch		ch collates after c as a 2-1 mapping
d	D,d	
e	E,È,É,Ê,Ë, e,è,é,ê,ë	
f	F,f	
g	G,g	
h	H,h	
i	I,Ì,Í,Î,Ï, i,ì,í,î,ï	
j	J,j	
k	K,k	
l	L,l	
m	M,m	
n	N,n	
o	O,Ò,Ó,Ô,Õ,Ö, o,ò,ó,ô,õ,ö	
p	P,p	
q	Q,q	
r	R,r	
s	S,s	
β		β collates equally with ss as a 1-2 mapping
t	T,t	
u	U,Ù,Ú,Û,Ü, u,ù,ú,û,ü	
v	V,v	
w	W,w	
x	X,x	
y	Y,Ý,y,ý	
z	Z,z	

3.5.5 Pseudo-Language 5 (VSX4L0)

There are no collating information requirements for pseudo-language `VSX4L0`, as it is not used by the collation tests. Typically it is either set up with no collating sequence definition or with the same collating sequence as the C locale.