

1 / *X/Open Technical Study*

2 **Internationalisation of X/Open Specifications:**
3 **Special Draft Sept 1995 for XTP TWG review**

4 *X/Open Company Ltd.*

5



6

© *December 1994, X/Open Company Limited*

7

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system,
8 or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or
9 otherwise, without the prior permission of the copyright owners.

9

10

X/Open Technical Study

11

Internationalisation of X/Open Specifications: Special Draft Sept 1995 for XTP TWG review

12

ISBN: 1-85912-080-6

13

X/Open Document Number: E408

14

Published by X/Open Company Ltd., U.K.

15

Any comments relating to the material contained in this document may be submitted to X/Open
16 at:

16

17

X/Open Company Limited

18

Apex Plaza

19

Forbury Road

20

Reading

21

Berkshire, RG1 1AX

22

United Kingdom

23

or by Electronic Mail to:

24

XoSpecs@xopen.org

Contents

25

26	Part	1	Internationalisation Overview	1
27	Chapter	1	Introduction.....	3
28	Chapter	2	Internationalisation	5
29		2.1	Overview	5
30		2.2	Character Sets and Encodings.....	6
31		2.3	The C Programming Language.....	9
32		2.4	Internationalisation Support in POSIX	11
33		2.5	Internationalisation Support in the X/Open CAE.....	12
34		2.5.1	XPG4 Facilities.....	12
35		2.5.2	Distributed Internationalisation Requirements	12
36		2.6	Current Work.....	14
37		2.6.1	Revision of the DISS.....	14
38		2.6.2	Definition and Registration of Locales.....	14
39		2.6.3	Complex Text Languages.....	14
40		2.6.4	Use of UNICODE/ISO 10646.....	15
41		2.6.5	Testing of Internationalised Components	15
42		2.6.6	Distributed Internationalisation Framework.....	15
43	Part	2	X/Open Interworking Specifications.....	17
44	Chapter	3	Introduction.....	19
45	Chapter	4	Potential Issues for Interworking.....	21
46	Chapter	5	Interworking Specifications.....	23
47		5.1	The XTI Specification	23
48		5.1.1	Overview	23
49		5.1.2	Internationalisation Implications	23
50		5.2	The XMPTN Specification.....	25
51		5.2.1	Overview	25
52		5.2.2	Internationalisation Implications	25
53		5.3	The XAP, XAP-TP and XAP-ROSE Specifications	26
54		5.3.1	Overview	26
55		5.3.2	Internationalisation Implications	26
56		5.4	The XOM Specification.....	27
57		5.4.1	Overview	27
58		5.4.2	Internationalisation Implications	28
59		5.5	The XFTAM Specification	29
60		5.5.1	Overview	29

61	5.5.2	Internationalisation Implications	29
62	5.6	The BSFT Specification	30
63	5.6.1	Overview	30
64	5.6.2	Internationalisation Implications	30
65	5.7	The X.400 API Specification.....	32
66	5.7.1	Overview	32
67	5.7.2	Internationalisation Implications	32
68	5.8	The XMS Specification	33
69	5.8.1	Overview	33
70	5.8.2	Internationalisation Implications	33
71	5.9	The XDS Specification.....	34
72	5.9.1	Overview	34
73	5.9.2	Internationalisation Implications	34
74	5.10	The XNFS Specification	36
75	5.10.1	Overview	36
76	5.10.2	Internationalisation Implications	36
77	5.11	The (PC)NFS Specification.....	39
78	5.11.1	Overview	39
79	5.11.2	Internationalisation Implications	39
80	5.12	The SMB Protocols Specification	40
81	5.12.1	Overview	40
82	5.12.2	Internationalisation Implications	40
83	5.13	The IPC Mechanisms for SMB Specification.....	42
84	5.13.1	Overview	42
85	5.13.2	Internationalisation Implications	42
86	Chapter 6	Conclusions and Recommendations.....	43
87	Chapter 7	Change Requests for Internationalisation	45
88	7.1	The XTI Specification	46
89	7.2	The XAP, XAP-TP and XAP-ROSE Specifications	48
90	7.3	The XOM Specification	50
91	7.4	The BSFT Specification	52
92	7.5	The XFTAM Specification	53
93	7.6	The X.400 API Specification.....	54
94	7.7	The XDS Specification.....	55
95	7.8	The XNFS Specification	57
96	7.9	The (PC)NFS Specification.....	57
97	7.10	The SMB Protocols Specification	57
98	7.11	The IPC Mechanisms for SMB Specification.....	57

99	Part	3	X/Open Data Management Specifications	59
100	Chapter	8	Introduction.....	61
101	Chapter	9	Structured Query Language (SQL).....	63
102		9.1	Overview	63
103		9.2	Multiple Character Sets	64
104		9.3	Use of Standard Names	64
105		9.4	Character Set Not Determined by Locale	65
106		9.5	Encodings	65
107		9.6	String Operations.....	67
108		9.7	ISO 10646 and C	67
109		9.8	Reserved Words and Special Characters	68
110		9.9	Numeric and Date Literals.....	68
111		9.10	Diagnostic Information.....	68
112		9.11	Arithmetical Expressions	69
113		9.12	Directionality	69
114	Chapter	10	Data Management Specifications	71
115		10.1	The SQL Specification	71
116		10.1.1	Overview	71
117		10.1.2	Internationalisation Implications	71
118		10.2	The CLI Specification	72
119		10.2.1	Overview	72
120		10.2.2	Internationalisation Implications	72
121		10.3	The RDA Specification.....	74
122		10.3.1	Overview	74
123		10.3.2	Internationalisation Implications	74
124	Chapter	11	Conclusions and Recommendations.....	77
125		11.1	Conclusions.....	77
126		11.2	Recommendations	78
127	Part	4	X/Open DTP Specifications	81
128	Chapter	12	Introduction.....	83
129	Chapter	13	DTP Specifications	85
130		13.1	The TX (Transaction Demarcation) Specification	85
131		13.1.1	Overview	85
132		13.1.2	Internationalisation Implications	85
133		13.2	The XA Specification	87
134		13.2.1	Overview	87
135		13.2.2	Internationalisation Implications	87
136		13.3	The XA+ Specification	89
137		13.3.1	Overview	89
138		13.3.2	Internationalisation Implications	89

139	Chapter 14	Conclusions and Recommendations.....	91
140	14.1	Summary	91
141	14.2	Function Names, Arguments, Characteristics and Return Codes....	91
142	14.3	ISO C and Common Usage C.....	92
143	Chapter 15	Change Requests for Internationalisation	93
144	15.1	The TX (Transaction Demarcation) Specification	94
145	15.2	The XA Specification	95
146	15.3	The XA+ Specification	98
147	Part 5	X/Open Systems Management Specifications	103
148	Chapter 16	Introduction.....	105
149	Chapter 17	Systems Management Specifications	107
150	17.1	The XMP Specification.....	107
151	17.1.1	Overview	107
152	17.1.2	Internationalisation Implications	107
153	17.2	The XMPP Specification	109
154	17.2.1	Overview	109
155	17.2.2	Internationalisation Implications	109
156	17.3	The XGDMO Specification	110
157	17.3.1	Overview	110
158	17.3.2	Internationalisation Implications	110
159	17.4	The UMA Specifications.....	111
160	17.4.1	Overview	111
161	17.4.2	Internationalisation Implications	112
162	17.5	The XBSA Specification	115
163	17.5.1	Overview	115
164	17.5.2	Internationalisation Implications	115
165	17.6	The XSMS Specification.....	116
166	17.6.1	Overview	116
167	17.6.2	Internationalisation Implications	116
168	Chapter 18	Conclusions and Recommendations.....	119
169	18.1	Conclusions.....	119
170	18.1.1	Character String Identifiers	119
171	18.1.2	Null-Terminated Strings.....	120
172	18.1.3	Descriptive Text	120
173	18.1.4	String Comparisons.....	121
174	18.1.5	Input and Output Character Sets	121
175	18.1.6	Use of specific Date/Time Formats	122
176	18.2	Recommendations	123
177	18.2.1	Character String Identifiers	123
178	18.2.2	Null-Terminated Strings.....	123
179	18.2.3	Descriptive Text	124
180	18.2.4	String Comparisons.....	124
181	18.2.5	Input and Output Character Sets	124

182	Chapter	19	Change Requests for Internationalisation	125
183		19.1	The XMP Specification.....	126
184		19.2	The XGDMO Specification	128
185		19.3	The UMA Specifications.....	129
186		19.4	The XBSA Specification	135
187		19.5	The XSMS Specification.....	137
188	Part	6	Glossary and Index	139
189			Glossary	141
190			Index.....	145

191

192

X/Open193
194
195
196

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

197
198
199
200
201
202
203

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

204
205
206
207
208

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

209
210
211

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

212

X/Open Technical Publications213
214
215

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

216

There are two types of X/Open specification:

217

- *CAE Specifications*

218
219
220

CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

221
222
223
224
225

Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

226
227
228
229

CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

230 • *Preliminary Specifications*

231 These specifications, which often address an emerging area of technology and consequently
 232 are not yet supported by multiple sources of stable conformant implementations, are
 233 released in a controlled manner for the purpose of validation through implementation of
 234 products. A Preliminary specification is not a draft specification. In fact, it is as stable as
 235 X/Open can make it, and on publication has gone through the same rigorous X/Open
 236 development and review procedures as a CAE specification.

237 Preliminary specifications are analogous to the *trial-use* standards issued by formal standards
 238 organisations, and product development teams are encouraged to develop products on the
 239 basis of them. However, because of the nature of the technology that a Preliminary
 240 specification is addressing, it may be untried in multiple independent implementations, and
 241 may therefore change before being published as a CAE specification. There is always the
 242 intent to progress to a corresponding CAE specification, but the ability to do so depends on
 243 consensus among X/Open members. In all cases, any resulting CAE specification is made as
 244 upwards-compatible as possible. However, complete upwards-compatibility from the
 245 Preliminary to the CAE specification cannot be guaranteed.

246 In addition, X/Open publishes:

247 • *Guides*

248 These provide information that X/Open believes is useful in the evaluation, procurement,
 249 development or management of open systems, particularly those that are X/Open-
 250 compliant. X/Open Guides are advisory, not normative, and should not be referenced for
 251 purposes of specifying or claiming X/Open conformance.

252 • *Technical Studies*

253 X/Open Technical Studies present results of analyses performed by X/Open on subjects of
 254 interest in areas relevant to X/Open's Technical Programme. They are intended to
 255 communicate the findings to the outside world and, where appropriate, stimulate discussion
 256 and actions by other bodies and the industry in general.

257 • *Snapshots*

258 These provide a mechanism for X/Open to disseminate information on its current direction
 259 and thinking, in advance of possible development of a Specification, Guide or Technical
 260 Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A
 261 Snapshot represents the interim results of an X/Open technical activity. Although at the time
 262 of its publication, there may be an intention to progress the activity towards publication of a
 263 Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no
 264 commitment regarding future development and further publication. Similarly, a Snapshot
 265 does not represent any commitment by X/Open members to develop any specific products.

266 **Versions and Issues of Specifications**

267 As with all *live* documents, CAE Specifications require revision, in this case as the subject
 268 technology develops and to align with emerging associated international standards. X/Open
 269 makes a distinction between revised specifications which are fully backward compatible and
 270 those which are not:

- 271 • a new *Version* indicates that this publication includes all the same (unchanged) definitive
 272 information from the previous publication of that title, but also includes extensions or
 273 additional information. As such, it *replaces* the previous publication.

- 274 • a new *Issue* does include changes to the definitive information contained in the previous
275 publication of that title (and may also include extensions or additional information). As such,
276 X/Open maintains *both* the previous and new issue as current publications.

277 **Corrigenda**

278 Most X/Open publications deal with technology at the leading edge of open systems
279 development. Feedback from implementation experience gained from using these publications
280 occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to
281 reported problems are communicated by means of Corrigenda.

282 The reader of this document is advised to check periodically if any Corrigenda apply to this
283 publication. This may be done either by email to the X/Open info-server or by checking the
284 Corrigenda list in the latest X/Open Publications Price List.

285 To request Corrigenda information by email, send a message to info-server@xopen.co.uk with
286 the following in the Subject line:

287 request corrigenda; topic index

288 This will return the index of publications for which Corrigenda exist.

289 **This Document**

290 This document is a Technical Study (see above). It identifies the implications of
291 internationalisation requirements on the following types of X/Open specification:

- 292 • Interworking Specifications
- 293 • Data Management Specifications
- 294 • Distributed Transaction Processing (DTP) Specifications
- 295 • Systems Management Specifications.

296 The structure of this technical study is as follows:

- 297 • Part 1 — Internationalization Overview
 - 298 — Chapter 1 is an introduction to this technical study.
 - 299 — Chapter 2 discusses the subject of internationalisation in general terms and describes the
300 provisions that have been made for it in international standards and in the X/Open CAE.
- 301 • Part 2 — X/Open Interworking Specifications
 - 302 — Chapter 3 introduces the X/Open interworking specifications that are considered by this
303 technical study.
 - 304 — Chapter 4 describes the criteria that have been followed in identifying internationalisation
305 issues in the X/Open interworking specifications.
 - 306 — Chapter 5 analyses the implications of internationalisation on the X/Open interworking
307 specifications.
 - 308 — Chapter 6 presents conclusions and recommendations.
 - 309 — Chapter 7 contains a set of proposed *internationalisation* Change Requests (CRs) for the
310 X/Open interworking specification.
- 311 • Part 3 — X/Open Data Management Specifications
 - 312 — Chapter 8 introduces the X/Open data management specifications that are considered by
313 this technical study.

- 314 — Chapter 9 discusses general internationalisation issues that are associated with SQL.
- 315 — Chapter 10 analyses the implications of internationalisation on the X/Open data
- 316 management specifications.
- 317 — Chapter 11 presents conclusions and recommendations.
- 318 • Part 4 — X/Open DTP Specifications
- 319 — Chapter 12 introduces the X/Open Distributed Transaction Processing (DTP)
- 320 specifications that are considered by this technical study.
- 321 — Chapter 13 analyses the implications of internationalisation on the X/Open DTP
- 322 specifications.
- 323 — Chapter 14 presents conclusions and recommendations.
- 324 — Chapter 15 contains a set of proposed *internationalisation* Change Requests (CRs) for the
- 325 X/Open DTP specification.
- 326 • Part 5 — X/Open Systems Management Specifications
- 327 — Chapter 16 introduces the X/Open systems management specifications that are
- 328 considered by this technical study.
- 329 — Chapter 17 analyses the implications of internationalisation on the X/Open systems
- 330 management specifications.
- 331 — Chapter 18 presents conclusions and recommendations.
- 332 — Chapter 19 contains a set of proposed *internationalisation* Change Requests (CRs) for the
- 333 X/Open systems management specification.
- 334 • Part 6 provides a glossary and an index.

335 **Intended Audience**

336 This technical study is aimed in general at all users and suppliers who wish to understand the
 337 issues surrounding internationalisation and how these can be solved. In particular, it is aimed at
 338 implementors and application developers who use X/Open's interworking, data management,
 339 DTP and systems management specifications.

340 **Typographical Conventions**

341 The following typographical conventions are used throughout this document:

- 342 • **Bold** font is used in text for filenames, keywords, type names, data structures and their
- 343 members.
- 344 • *Italic* strings are used for emphasis or to identify the first instance of a word requiring
- 345 definition. Italics in text also denote:
 - 346 — variable names, for example, substitutable argument prototypes and environment
 - 347 variables
 - 348 — C-language functions; these are shown as follows: *name()*
- 349 • Normal font is used for the names of constants and literals.
- 350 • The notation **<file.h>** indicates a C-language header file.
- 351 • Names surrounded by braces, for example, {ARG_MAX}, represent symbolic limits or
- 352 configuration values, which may be declared in appropriate C-language header files by

Preface

- 353 means of the C **#define** construct.
- 354 • The notation [ABCD] is used to identify a coded return value in C.
- 355 • Syntax and code examples are shown in *fixed width font*.
- 356 • Variables within syntax statements are shown in *italic fixed width font*.

357

Trade Marks

358
359

UNIX[®] is a registered trade mark in the United States and other countries, licensed exclusively through X/Open Company Limited.

360
361

X/Open[®] is a registered trade mark, and the “X” device is a trade mark, of X/Open Company Limited.

Referenced Documents

362

The following standards are referenced in this technical study:

363 ANSI C

364 American National Standard for Information Systems: standard X3.159-1989, Programming
365 Language C.
366

367 ASN.1

368 ISO 8824: 1990 Information Technology — Open Systems Interconnection — Specification of
369 Abstract Syntax Notation One (ASN.1).

370 ASN.1 DIS

371 ISO DIS 8824: 1992-1993 Information Technology — Open Systems Interconnection —
372 Abstract Syntax Notation One (ASN.1) — Specification of Basic Notation.

373 CMISP

374 ISO/IEC 9596-1: 1991, Common Management Information Service Protocol.

375 FIPS 127-2

376 US National Institute of Standards and Technology (NIST), Federal Information Processing
377 Standard, FIPS 127-2, Database Language SQL.

378 GDMO

379 ISO/IEC 10165-4:1992, Information Technology — Open Systems Interconnection —
380 Structure of Management Information — Part 4: Guidelines for the Definition of Managed
381 Objects.

382 ISO/IEC 646

383 ISO/IEC 646: 1991, Information Processing — ISO 7-bit Coded Character Set for Information
384 Interchange.

385 ISO 2022

386 ISO 2022: 1986 Information Processing — ISO 7-bit and 8-bit Coded Character Sets — Coded
387 Extension Techniques.

388 ISO 2375

389 ISO 2375: 1985 Data Processing — Procedure for Registration of Escape Sequences.

390 ISO 3166

391 ISO 3166: 1988, Codes for the Representation of Names of Countries, Bilingual edition.

392 ISO/IEC 8571

393 ISO/IEC 8571, Information Processing Systems — Open Systems Interconnection — File
394 Transfer, Access and Management.

395 Part 1: General Introduction (1988)

396 Part 2: Virtual Filestore Definition (1988)

397 Part 3: File Service Definition (1988)

398 Part 4: File Protocol Specification (1988).

399 ISO 8859

400 ISO 8859: 1987 Information Processing — 8-bit Single-byte Coded Graphic Character

- 401 Part 1, Latin Alphabet No. 1 (1987)
 402 Part 2, Latin Alphabet No. 2 (1987)
 403 Part 3, Latin Alphabet No. 3 (1988)
 404 Part 4, Latin Alphabet No. 4 (1988)
 405 Part 5, Latin/Cyrillic Alphabet (1988)
 406 Part 6, Latin/Arabic Alphabet (1987)
 407 Part 7, Latin/Greek Alphabet (1987)
 408 Part 8, Latin/Hebrew Alphabet (1988)
 409 Part 9, Latin Alphabet No. 5 (1989).
- 410 ISO/IEC 9594
 411 ISO/IEC 9594:1990, Information Technology — Open Systems Interconnection — The
 412 Directory, Parts 1 to 8:
- 413 Part 1: Overview of Concepts, Models and Services (1990)
 414 Part 2: Models (1990)
 415 Part 3: Abstract Service Definition (1990)
 416 Part 4: Procedures for Distributed Operation (1990)
 417 Part 5: Protocol Specifications (1990)
 418 Part 6: Selected Attribute Types (1990)
 419 Part 7: Selected Object Classes (1990)
 420 Part 8: Authentication Framework (1990)
- 421 ISO/IEC 10021
 422 ISO/IEC 10021, Information Processing Systems — Text Communication — Message
 423 Oriented Text Interchange System:
- 424 Part 1: System and Service Overview (1990)
 425 Part 2: Overall Architecture (1990)
 426 Part 3: Abstract Service Definition Conventions (1990)
 427 Part 4: Message Transfer System: Abstract Service
 428 Definition and Procedures (1990)
 429 Part 5: Message Store: Abstract Service Definition (1990)
 430 Part 6: Protocol Specifications (1990)
 431 Part 7: Interpersonal Messaging System (1990).
- 432 ISO/IEC ISP 10607-2
 433 ISO/IEC ISP 10607-2:1990, Information Technology — International Standardized Profiles
 434 AFTnn — File Transfer, Access and Management — Part 2: Definition of Document Types,
 435 Constraint Sets and Syntaxes.
- 436 ISO/IEC 10646
 437 ISO/IEC 10646-1:1993, Information Technology — Universal Multiple-Octet Coded
 438 Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.
- 439 ISO C
 440 ISO/IEC 9899:1990, Programming Languages — C (technically identical to ANSI standard
 441 X3.159-1989).
- 442 ISO MSE
 443 ISO/IEC 9899:1990 Amendment 1:1994, Multibyte Support Extensions for ISO C.
- 444 ISO RDA
 445 RDA Generic
 446 ISO/IEC 9579-1:1993, Information Technology — Open Systems Interconnection —
 447 Remote Database Access — Part 1: Generic Model, Service, and Protocol.

Referenced Documents

- 448 RDA SQL Specialization
449 ISO/IEC 9579-2: 1993, Information Technology — Open Systems Interconnection —
450 Remote Database Access — Part 2: SQL Specialization.
- 451 ISO SQL
452 ISO/IEC 9075:1992, Information Technology — Database Language SQL (technically
453 identical to ANSI standard X3.135-1992).
- 454 POSIX.1
455 ISO/IEC 9945-1: 1990 (also IEEE Std. 1003.1:1990), Portable Operating System (POSIX) —
456 Part 1: System Application Program Interface.
- 457 POSIX.2
458 IEEE Std. 1003.2: 1992, Portable Operating System (POSIX) — Part 2: Shell and Utilities.
- 459 SNMP
460 Internet RFC 1157, The Simple Network Management Protocol.
- 461 SQL3
462 ISO-ANSI Working Draft — Database Language SQL, May 1993.
- 463 T.61
464 CCITT Recommendation T.61: 1984, Character Repertoire and Coded Character Sets for the
465 International Teletex Service.
- 466 T.100
467 CCITT Recommendation T.100: 1984, International Information Exchange for Interactive
468 Videotex.
- 469 UNICODE
470 The Unicode Consortium, The Unicode Standard, Worldwide Character Encoding Version
471 1.0, Volume One, Addison-Wesley, 1991.
- 472 X.400
473 CCITT Recommendations X.400-X.420: 1988, Data Communications Networks — Message
474 Handling Systems. These recommendations are technically aligned with ISO 10021.
- 475 X.500
476 CCITT Recommendations X.500-X.521: 1988, Data Communications Networks — Directory.
477 These recommendations are technically aligned with ISO 9594.
- 478 The following X/Open documents are referenced in this technical study:
- 479 BSFT
480 X/Open CAE Specification, December 1991, Byte Stream File Transfer (BSFT)
481 (ISBN: 1-872630-27-8, C194).
- 482 CLI
483 X/Open Snapshot, October 1992, Data Management: SQL Call Level Interface (CLI)
484 (ISBN 1-872630-63-4, S203).
- 485 CORBA
486 X/Open CAE Specification, August 1994, Common Object Request Broker: Architecture &
487 Specification, (ISBN: 1-85912-044-X, C432).
- 488 CPI-C, Version 2
489 X/Open CAE Specification, October 1995, Distributed Transaction Processing: The CPI-C
490 Specification, Version 2 (ISBN: 1-85912-135-7, C419).

- 491 DISS, Issue 1
 492 X/Open Snapshot, November 1992, Distributed Internationalisation Services
 493 (ISBN: 1-872630-75-8 S213).
- 494 DTP
 495 X/Open Guide, November 1993, Distributed Transaction Processing: Reference Model,
 496 Version 2 (ISBN: 1-85912-019-9, G307).
- 497 FSS-UTF
 498 X/Open Preliminary Specification, May 1993, File System Safe UCS Transformation Format
 499 (FSS-UTF) (ISBN: 1-872630-96-0, P316).
- 500 Internationalisation Guide
 501 X/Open Guide, July 1993, Internationalisation Guide, Version 2 (ISBN: 1-859120-02-4, G304).
- 502 Interworking Internationalization
 503 X/Open Snapshot, May 1993, Internationalization of Interworking Specifications
 504 (ISBN 1-872630-87-1, S302).
- 505 IPC SMB
 506 X/Open Developers' Specification — IPC Mechanisms for SMB, XO/CAE/91/500, The
 507 X/Open Co. Ltd, 1991.
- 508 Layout Services
 509 X/Open Snapshot, December 1994, Portable Layout Services: Context-dependent and
 510 Directional Text (ISBN: 1-85912-075-X, S425).
- 511 Locale Registry
 512 X/Open Electronic Publication, October 1993, Locale Registry Procedures (ISBN: 1-872630-
 513 94-4, G303).
- 514 Migration Guide
 515 X/Open Guide, July 1992, XPG3-XPG4 Base Migration Guide (ISBN: 1-872630-49-9, G204).
- 516 (PC)NFS
 517 X/Open Developers' Specification, August 1990, Protocols for X/Open PC Interworking:
 518 (PC)NFS (ISBN: 1-872630-00-6, D030 or XO/DEV/90/030).
- 519 RDA
 520 X/Open CAE Specification, August 1993, Data Management: SQL Remote Database Access
 521 (ISBN: 1-872630-98-7, C307).
- 522 SMB
 523 X/Open CAE Specification, October 1992, Protocols for X/Open PC Interworking: SMB,
 524 Version 2 (ISBN: 1-872630-45-6, C209).
- 525 SQL
 526 X/Open CAE Specification, August 1992, Structured Query Language (SQL)
 527 (ISBN: 1-872630-58-8, C201).
- 528 TX
 529 X/Open Preliminary Specification, October 1992, Distributed Transaction Processing: The
 530 TX (Transaction Demarcation) Specification (ISBN: 1-872630-65-0, P209).
- 531 TxRPC
 532 X/Open Preliminary Specification, July 1993, Distributed Transaction Processing: The
 533 TxRPC Specification (ISBN: 1-85912-000-8, P305).
- 534 UCS
 535 X/Open Technical Study, February 1994, Universal Multiple-Octet Coded Character Set

Referenced Documents

- 536 Coexistence and Migration (ISBN: 1-85912-031-8, E401).
- 537 UMA
- 538 Forthcoming X/Open Guide, expected to be published in 1995, Guide to Universal
- 539 Measurement Architecture (UMA), (ISBN 1-85912-073-3, G414).
- 540 The version of UMA used during this technical study was X/Open's pre-publication draft
- 541 dated October 6, 1994.
- 542 UMA DCI
- 543 Forthcoming X/Open Preliminary Specification, expected to be published in 1995, Systems
- 544 Management: UMA Data Capture Interface, (ISBN 1-85912-068-7, P434).
- 545 The version of UMA DCI used during this technical study was X/Open's pre-publication
- 546 draft dated November 1, 1994.
- 547 UMA DPD
- 548 Forthcoming X/Open Preliminary Specification, expected to be published in 1995, Systems
- 549 Management: UMA Data Pool Definitions, (ISBN 1-85912-069-5, P435).
- 550 The version of UMA DPD used during this technical study was X/Open's pre-publication
- 551 draft dated November 7, 1994.
- 552 UMA MLI
- 553 Forthcoming X/Open Preliminary Specification, expected to be published in 1995, Systems
- 554 Management: UMA Measurement Layer Interface, (ISBN 1-85912-072-5, P426).
- 555 The version of UMA MLI used during this technical study was X/Open's pre-publication
- 556 draft dated November 7, 1994.
- 557 XA
- 558 X/Open CAE Specification, December 1991, Distributed Transaction Processing: The XA
- 559 Specification (ISBN: 1-872630-24-3, C193 or XO/CAE/91/300).
- 560 XA+
- 561 X/Open Snapshot, July 1994, Distributed Transaction Processing: The XA+ Specification,
- 562 Version 2 (ISBN: 1-85912-046-6, S423).
- 563 XAP PS
- 564 X/Open Preliminary Specification, June 1992, ACSE/Presentation Services API (XAP)
- 565 (ISBN: 1-872630-53-7, P203).
- 566 XAP
- 567 X/Open CAE Specification, September 1993, ACSE/Presentation Services API (XAP) (ISBN:
- 568 1-872630-91-X, C303).
- 569 XAP-ROSE
- 570 X/Open Preliminary Specification, December 1993, Remote Operations Service Element
- 571 (XAP-ROSE) API, (ISBN 1-872630-86-3, P302).
- 572 XAP-TP
- 573 X/Open Preliminary Specification, February 1994, ACSE/Presentation: Transaction
- 574 Processing API (XAP-TP) (ISBN: 1-872630-85-5, P216).
- 575 XATMI
- 576 X/Open Preliminary Specification, July 1993, Distributed Transaction Processing: The
- 577 XATMI Specification (ISBN: 1-872630-99-5, P306).
- 578 XBSA
- 579 Forthcoming X/Open Preliminary Specification, expected to be published in 1995, Systems

- 580 Management: Backup Services API (XBSA), (ISBN 1-85912-056-3, P424).
581 The version of XBSA used during this technical study was X/Open's pre-publication draft
582 dated October 7, 1994.
- 583 XDS
584 X/Open CAE Specification, November 1991, API to Directory Services (XDS)
585 (ISBN: 1-872630-18-9, C190 or XO/CAE/91/090).
- 586 XDS, Issue 2
587 X/Open CAE Specification, February 1994, API to Directory Services (XDS), Issue 2
588 (ISBN: 1-85912-007-5, C317).
- 589 XFTAM PS
590 X/Open Preliminary Specification, September 1992, FTAM High-level API (XFTAM)
591 (ISBN: 1-872630-60-X, P206).
- 592 XFTAM
593 X/Open CAE Specification, January 1994, FTAM High-level API (XFTAM)
594 (ISBN: 1-85912-010-5, C304).
- 595 XGDMO
596 X/Open Preliminary Specification, March 1994, Systems Management: GDMO to XOM
597 Translation Algorithm (ISBN: 1-85912-023-7, P319).
- 598 XMP
599 X/Open CAE Specification, March 1994, Systems Management: Management Protocol API
600 (ISBN 1-85912-027-X, C306).
- 601 XMPP
602 X/Open CAE Specification, October 1993, Systems Management: Management Protocol
603 Profiles (ISBN 1-85912-018-0, C206).
- 604 XMPTN Access Node
605 X/Open Preliminary Specification, September 1994, Multiprotocol Transport Networking
606 (MPTN): Access Node, (ISBN 1-85912-040-7, P408).
- 607 XMPTN Address Mapper
608 X/Open Preliminary Specification, September 1994, Multiprotocol Transport Networking
609 (MPTN): Address Mapper (ISBN 1-85912-039-3, P407).
- 610 XMS
611 X/Open CAE Specification, June 1993, Message Store API (XMS) (ISBN: 1-872630-83-9,
612 C305).
- 613 XSMS
614 Forthcoming X/Open Preliminary Specification, expected to be published in 1995, Systems
615 Management: Management Services in an OMG Environment, (ISBN 1-85912-047-4, P421).
616 The version of XSMS used during this technical study was X/Open's pre-publication draft
617 dated November 1, 1994.
- 618 XNFS
619 X/Open CAE Specification, October 1992, Protocols for X/Open Interworking: XNFS, Issue
620 4 (ISBN: 1-872630-66-9, C218).
- 621 XOM
622 X/Open CAE Specification, November 1991, OSI-Abstract-Data Manipulation API (XOM)
623 (ISBN: 1-872630-17-0, C180 or XO/CAE/91/080).

Referenced Documents

- 624 XOM, Issue 2
625 X/Open CAE Specification, February 1994, OSI-Abstract-Data Manipulation API (XOM),
626 Issue 2 (ISBN: 1-85912-008-3, C315).
- 627 XPG1
628 X/Open Portability Guide, July 1985 (ISBN: 0-444-87839-4).
- 629 XPG2
630 X/Open Portability Guide, five volumes, January 1987 (ISBN: 0-444-70179-6).
- 631 XPG3
632 X/Open Specification, 1988, 1989, February 1992 (ISBN: 1-872630-43-X, T921); this
633 specification was formerly X/Open Portability Guide, seven volumes, January 1989
634 (ISBN: 0-13-685819-8, XO/XPG/89/000).
- 635 XPG4
- 636 XBD, Issue 4
637 X/Open CAE Specification, July 1992, System Interface Definitions, Issue 4
638 (ISBN: 1-872630-46-4, C204).
- 639 XCU, Issue 4
640 X/Open CAE Specification, July 1992, Commands and Utilities, Issue 4 (ISBN:
641 1-872630-48-0, C203).
- 642 XSH, Issue 4
643 X/Open CAE Specification, July 1992, System Interfaces and Headers, Issue 4
644 (ISBN: 1-872630-47-2, C202).
- 645 XRM
646 X/Open Guide, August 1993, Systems Management: Reference Model (ISBN: 1-85912-05-9,
647 G207).
- 648 XTI
649 X/Open CAE Specification, January 1992, X/Open Transport Interface (XTI)
650 (ISBN: 1-872630-29-4, C196 or XO/CAE/91/600).
- 651 X.400 API
652 X/Open CAE Specification, December 1991, API to Electronic Mail (X.400)
653 (ISBN: 1-872630-19-7, C191 or XO/CAE/91/100).
- 654 X.400 API, Issue 2
655 X/Open CAE Specification, February 1994, API to Electronic Mail (X.400), Issue 2,
656 (ISBN: 1-85912-009-1, C316).

1 / *X/Open Technical Study*

2 **Part 1**

3 **Internationalisation Overview**

4 *X/Open Company Ltd.*

Introduction

5

6 Computer systems and applications are increasingly expected to work in an international
7 environment in which different languages, character sets and cultural conventions are in use.
8 This poses a number of requirements. The growth of distributed computing, with systems and
9 applications interworking across networks, is making these requirements more urgent. They
10 affect the networking technology on which distributed systems are based, and also the methods
11 by which data is stored, manipulated and administered on behalf of users. If computer systems
12 do not take the requirements of internationalisation into account, the ability of a distributed
13 system to work in different languages and cultural environments is limited.

14 This technical study identifies the implications of internationalisation requirements on the
15 following types of X/Open specification; each of which is discussed in more detail in the
16 following parts of this document:

- 17 • Interworking Specifications — see Part 2
- 18 • Data Management Specifications — see Part 3
- 19 • Distributed Transaction Processing (DTP) Specifications — see Part 4
- 20 • Systems Management Specifications — see Part 5.

21 Before these four parts, Chapter 2 discusses the general subject of internationalisation, and in
22 particular describes the provisions that are made for it in international standards and in the
23 X/Open Common Applications Environment (CAE).

2.1 Overview

Computer systems must meet the needs of users who speak different languages, conform to different cultural conventions and follow different business practices. This means that the facilities of the X/Open open systems environment must not impose constraints on the users' languages, cultural conventions or business practices, and must include facilities that support the development of applications that can be used in multiple language, cultural and business environments.

Understanding of the implications of this has evolved as the X/Open open systems environment has developed. It is evolving still.

The most obvious area in which constraints can be imposed, and the area that has received the most attention, is that of character sets and their encodings. However, programs have often imposed other constraints by making assumptions about:

- directionality (whether text is written from right to left or from left to right)
- collation rules used in comparing, ordering and sorting character strings
- rules for character classification into categories such as alphabetic, numeric, punctuation and so on
- shift rules for character case conversion
- the way in which numbers are written (for example, the use of a comma (,) or decimal-point (.) as separator)
- the value and positioning of the currency symbol
- the way in which dates are written (for example, dd/mm/yy or mm/dd/yy, or using Asian formats with dissimilar date component separators)
- the way in which times are written (for example, 10:24 PM, 22.24, 10h24)
- the use of upper and lower case characters
- the language of the user interface (for example, error messages in a particular natural language have often been hard-coded into a program).

This chapter summarises the provisions that have been made in international standards and in the X/Open open systems environment for addressing internationalisation issues. The international standards concerned fall into two categories. The first is that of standards for character sets and encodings used in data communication. The second is that of standards for information processing, specifically the C programming language and the POSIX operating system interface. X/Open publications have always taken account of developments in international standards, and have often anticipated and influenced them. This chapter therefore concludes with an indication of the current direction of internationalisation work within X/Open.

60 2.2 Character Sets and Encodings

61 A wide variety of character sets is used to represent the languages of the world. This document
62 is written in the English language, represented using the characters of the basic Latin alphabet.
63 Other Western European languages are represented using character sets that include those of the
64 basic Latin alphabet plus a few additional characters (different additional characters are used by
65 each language). Other languages (such as Greek and Russian) use character sets that are
66 alphabetic but are not variants of the Latin alphabet. Yet other languages, such as Japanese and
67 Chinese, use ideographic scripts that are not alphabetic. Mathematical and scientific text, in any
68 language, uses characters borrowed from several different alphabets.

69 When held in computer storage, and while being transmitted between computers, characters are
70 encoded as bit patterns. The bit patterns that constitute the encodings of a character set are
71 called a codeset.

72 A number of encoding schemes used to represent characters being transmitted between
73 computers have been standardised by national standards bodies, the CCITT (now the ITU-T)
74 and ISO. In each of these standards, a character is typically encoded as one or more octets,
75 where an octet is a sequence of 8 bits, each of which can take the value 0 or 1.

76 Early communication protocols were designed for communication over low bandwidth lines
77 and with relatively “dumb” devices such as teletypes. They used the minimum possible
78 number of bits per character, and distinguished between graphic characters, which would be
79 printed, and control characters, which would affect the operation of the remote device. Control
80 characters included characters used to control communication (such as the <SOH> character
81 that indicated the start of header information) and also characters used to control printing (such
82 as the <CR> character that, on a teletype, caused the carriage to return to its starting position).

83 These facts affected the character encoding schemes that were used in conjunction with early
84 protocols. The American Standard Code for Information Interchange (ASCII) was directly
85 descended from such schemes. It is the basis of the referenced ISO 646 international standard,
86 has considerably influenced later schemes, and is still in use. It represents the basic Latin
87 alphabet, plus some additional control characters, using 7 bits per character. Control characters
88 are encoded with values in the range 00 to 1F (hexadecimal).

89 Modern communication protocols, including the OSI protocols standardised by ISO and the
90 protocols of the Internet protocol suite, transport 8-bit data transparently. This allows the use of
91 encoding schemes that use all 8 bits of an octet and that do not reserve particular values for
92 protocol control purposes.

93 A mechanism, intended for use with 7-bit or 8-bit encoding schemes, by which several different
94 schemes can be used within a single transmission, is defined in the referenced ISO 2022
95 international standard. In this mechanism, certain control characters perform a *shift* function
96 which determines how subsequent codes are to be interpreted. (This is by analogy with a
97 typewriter, on which the <Shift> keys determine the symbols that will be printed when other
98 keys are subsequently pressed.) The mechanism also allows the possibility that the encoding of
99 a character can occupy more than one octet. Essentially, the *unshifted* codes represent the
100 characters of the basic Latin alphabet, while *shifted* codes represent the characters of some other
101 character set (as agreed by the communicating parties). With multiple-octet-per-character
102 encoding schemes, any character set can be encoded.

103 A register of character sets and encodings is defined in the referenced ISO 2375 international
104 standard. Encodings for most Western European character sets and for Japanese Kanji are
105 registered.

106 Encodings compatible with ISO 2022 for the character sets of most languages used in Europe
107 and North America (including Greenlandic, Russian and Turkish) and also of Afrikaans, Arabic,
108 Esperanto and Hebrew, are defined in the referenced ISO 8859 international standard.

109 Encoding schemes that use the mechanism of the referenced ISO 2022 international standard
110 have been standardised for use in the Teletex service (see the referenced T.61 CCITT
111 Recommendation) and the Videotex service (see the referenced T.100 CCITT Recommendation).

112 It should be noted that all the above standards use the same encodings as the referenced ISO 646
113 international standard for the characters of the basic Latin alphabet. They also maintain the
114 principle, even in multi-octet encodings, that octets in the range 00 to 1F (hexadecimal) are
115 reserved for control characters.

116 However, the latest encoding standard, ISO 10646 (which incorporates the work of the
117 UNICODE consortium), departs from these principles.

118 ISO 10646 is intended to cover the character sets of all languages that may be used in
119 conjunction with computer systems. It defines a four-octet representation for each character.
120 The characters whose representations have zero as their two most significant octets form what is
121 known as the Basic Multilingual Plane (this includes most alphabetic character sets). Two forms
122 of encoding are permitted:

123 UCS-2 This form applies where only characters in the Basic Multilingual plane are used. In it,
124 the encoding of a character consists of the two least significant octets of its four-octet
125 representation.

126 UCS-4 This form permits the encoding of any character. In it, the encoding of a character
127 consists of the whole of its four-octet representation.

128 In addition to the UCS-2 and UCS-4 forms, ISO 10646 allows a composite graphical symbol to be
129 represented by the encoding of a base character followed by the encodings of one or more
130 combining characters. For example, the <e with acute accent> graphical symbol can be
131 represented in UCS-4 by (hex) 00 00 00 65 00 00 03 01 which is the encoding for lower case letter
132 <e> followed by the encoding for a combining <acute accent>. This symbol can also be
133 represented in UCS-4 by (hex) 00 00 00 E9 which is the encoding for Latin small letter <e with
134 acute accent>. A composite graphical symbol can thus have more than one encoding in UCS-4
135 (and also, similarly, in UCS-2). ISO 10646 defines three conformance levels:

- 136 1. combining characters are not allowed
- 137 2. some combining characters are allowed for certain scripts, such as Arabic, Hebrew, Indic
138 and Thai
- 139 3. combining characters are allowed with no restrictions.

140 The combinations of the three conformance levels with the two encoding forms gives six
141 possible ways in which an implementation can support the referenced ISO 10646 international
142 standard; the referenced UNICODE standard is equivalent to just one of these ways: UCS-2
143 Level 3.

144 A degree of compatibility with ISO 646 is maintained, in that the characters encoded by ISO 646
145 are encoded by the ISO 10646 using the ISO 646 codes preceded by the appropriate number of
146 null octets (one in the UCS-2 form; three in the UCS-4 form). For example, upper case A of the
147 Latin alphabet is encoded as (hex) 41 by the ISO 646, and as (hex) 00 41 by ISO 10646.

148 However, the ISO 646 encoding of any control or graphic character can appear as the leading
149 octet of the encoding of a completely different character in the UCS-2 form of ISO 10646, or as
150 any of the three leading octets of an encoding of the UCS-4 form. For example, the ISO 646
151 encoding of the End of Text (<ETX>) character appears as an octet of the ISO 10646 encodings of
152 the characters of the Greek alphabet. This makes it hard to use the UCS-2 or UCS-4 encoding for
153 data transmitted using communication protocols that assign special meanings to ISO 10646
154 control codes.

155 Recognising that this is a problem, the referenced ISO 10646 international standard defines a
156 UCS Transformation Format (UTF). When applied to an ISO 10646 encoding, this algorithm
157 yields a 1, 2, 3 or 5 octet value that is guaranteed not to contain the ISO 646 encodings of any
158 control character, or of the <SPACE> or characters. Data encoded in accordance with the
159 referenced ISO 10646 international standard, and then transformed by a UTF, can safely be
160 transmitted using communication protocols that assign special meanings to ISO 646 control
161 codes.

162 The algorithm defined by ISO 10646 (known as UTF-1) does not prevent encodings from
163 containing the ISO 646 encoding of the slash character, (hex) 2F. This limits its use on POSIX-
164 compliant systems, where the slash character is used to delimit segments of pathnames of files.
165 (There are similar problems with many systems that are not POSIX-compliant.) A second UTF,
166 known as FSS-UTF or UTF-8, has therefore been defined by the X/Open-Uniform Joint
167 Internationalisation Group (JIG). In this UTF, an octet with bit 8 set to zero can only appear as
168 the single-octet representation of the identical ISO 646 encoding. As well as being safe for
169 transmission by common communication protocols, such data can safely be processed by
170 applications that handle file pathnames on POSIX-compliant systems.

171 Most current implementations use the UCS-2 form of encoding, because it is much more
172 economical in its use of storage. A further transformation, known as UTF-16 (or "shifted
173 UNICODE") has been defined to enable applications on such systems to use some of the
174 characters that can be represented in UCS-4 but not in UCS-2. It does this by using pairs of
175 UCS-2 code positions to represent UCS-4 characters.

176 A full discussion of the issues pertaining to the use of ISO 10646 in Open Systems is contained in
177 the referenced UCS technical study.

178 2.3 The C Programming Language

179 In internal machine storage, characters are held in bytes. A byte is a unit of machine storage
180 containing at least 8 bits, each of which can take the value 0 or 1.

181 Often, the same encodings are used for characters held in machine storage as are used for
182 characters in transmission.

183 The facilities of the programming language determine how characters held in machine storage
184 can be manipulated by applications programs. For applications within the X/Open open
185 systems environment, the most important programming language is C. The character handling
186 facilities of the C programming language are of great importance with regard to the
187 development of internationalised applications.

188 Early versions of the C programming language, such as that specified in the referenced X/Open
189 **XPG1** Portability Guide, assumed a character encoding scheme similar to ASCII. They defined a
190 **char** type such that a value of type **char** could be held in a single (8-bit) byte, and defined a
191 character string to be an array of type **char** terminated by a null character. Many applications
192 programs written using such versions of C use these facilities, and are not amenable to
193 internationalisation, since they cannot handle multi-byte character set encodings.

194 In the version of C standardised by ANSI, and subsequently by ISO, some of the issues
195 associated with internationalisation are addressed. The **char** type still has values that can be
196 represented as single bytes, and character strings are still null-terminated arrays of type **char**.
197 However, multi-byte character encodings are possible, and can be held in strings with several
198 elements of type **char** representing each character. Also, the type **wchar_t** is provided for multi-
199 byte character encodings. In the referenced ISO C international standard it is defined to be such
200 that its range of values can represent all codes for the largest supported character set.

201 A set of character and string handling functions that have arguments that are of type **wchar_t**
202 and related types are defined in the referenced ISO MSE addendum to ISO C. For example,
203 function *strcat()* has been used since the earliest days of C programming, but is unsuitable for
204 use in internationalised programs because it has arguments of type **char ***. This constrains the
205 language to be one that uses an 8-bit character set. Many languages use character sets that are
206 not representable using 8 bits. The ISO MSE addendum includes function *wscat()*, which takes
207 wide character code arguments (type **wchar_t ***) and can be used in place of *strcat()* in
208 internationalised programs.

209 Because strings are null-terminated, an encoding scheme used in conjunction with ISO C must
210 not produce a null byte except as the encoding of the null character. The UCS-4 and UCS-2
211 encoding schemes do not have this property; therefore, use of the C language **char** data type as
212 defined in the referenced ISO C international standard in conjunction with the coded character
213 set defined in the referenced ISO 10646 international standard is problematic.

214 In addition to permitting flexibility of character sets and encodings, ISO C specifies a *locale*
215 mechanism which can be used to enable applications programs to be written without making
216 assumptions about language and cultural conventions. ISO 9899 (the ISO C Standard) specifies
217 functions for handling characters, strings, date and time, and formatted input/output. The
218 behaviour of these functions is affected by the current locale. This can be set by the applications
219 program to reflect the language and cultural environment in which the application is executing.
220 Application programs can also examine the current locale and modify their behaviour
221 accordingly.

222 Character collation, classification and case conversion, and the format of numbers, monetary
223 values and dates may all be affected by the locale. The ISO C standard does not prescribe
224 precisely how they are affected in any particular language and cultural environment (other than
225 a basic default environment); it just specifies a general mechanism whose use is
226 implementation-defined.

227 2.4 Internationalisation Support in POSIX

228 The locale mechanism of ISO C is extended by the referenced POSIX.1 international standard¹.
 229 This provides a means whereby an application program can use a locale that has been
 230 established in its process environment. For example, this allows a system to be shipped with a
 231 repertoire of pre-defined locales. The user or system administrator selects the locales in which
 232 applications run. However, POSIX.1 still specifies the general mechanism only, and contains no
 233 standardised descriptions of specific locales (other than the default locale).

234 Also, POSIX.1 defines a Portable Filename Character Set, which it recommends for use in
 235 international applications. (It allows other characters to be used in filenames, but advises that
 236 such names are not portable between different language and cultural environments). This
 237 consists of the upper and lower case characters of the Latin alphabet as used in English, the
 238 digits 0-9 and the period, underscore and hyphen characters (as found in ISO 646).

239 The interface specified in the referenced POSIX.2 IEEE standard² provides for a system to
 240 support multiple locales and, optionally, to allow the user to define locales. The behaviour of
 241 the system utilities is affected by the currently established locale. For example, the *ls* utility lists
 242 files, sorted by name according to the collation sequence in the current locale.

243 The current locale also affects certain aspects of the command interpreter (*sh*), although the
 244 reserved words that have special meaning are all defined using a particular character set - the
 245 Portable Character Set - that is required to be present in every supported locale. This Portable
 246 Character Set is a superset of the Portable Filename Character Set defined in the referenced
 247 POSIX.1 international standard. It includes additional punctuation characters such as { and }.

248 Several of the utilities defined in the referenced POSIX.2 IEEE standard can handle character-
 249 patterns called *regular expressions*. The meaning of “regular expression” is defined in terms of
 250 the current locale. For example, it is possible to specify the range of characters [a-z] as a regular
 251 expression; this would include the e-acute character in a French locale but not in an English one.

252 The definition of a locale includes the specification of an encoding of its characters. Stateless,
 253 but not stateful, multi-byte encodings are supported³.

254

-
- 255 1. The ISO 9945-1:1990 POSIX.1 standard is identical to IEEE Standard 1003.1-1990. It specifies a programming interface to
 256 operating system services.
- 257 2. IEEE 1003.2-1992 specifies a user interface to operating system services (commands and utilities).
- 258 3. A stateful encoding is one in which a code can set the interpreter into a state that affects the meaning of subsequent codes. An
 259 example of a stateful encoding is one that has a shift-lock code that causes subsequent codes for lower-case letters to be
 260 interpreted as the corresponding upper-case letters.

261 **2.5 Internationalisation Support in the X/Open CAE**

262 The need for internationalisation was stated in the first issue of the X/Open Portability Guide
263 (**XPG1**). A trial-use definition of facilities to enable internationalised applications programs to
264 be developed was contained in the second issue (**XPG2**). Issue 3 (**XPG3**) included some
265 mandatory facilities for the X/Open System Interface (XSI), which were largely aligned with the
266 internationalisation facilities of the POSIX.1 standard and the referenced ANSI C standard. They
267 were expanded and refined in Issue 4 (the referenced **XPG4-XSH X/Open CAE** specification)
268 including full conformance with ISO C. (ISO C is based on, and technically equivalent to, ANSI
269 C.)

270 A more complete description of the development of internationalisation facilities can be found in
271 the referenced X/Open **Internationalisation Guide**. The differences between Issue 3 and Issue 4
272 of the XSI are summarised in the referenced **Migration Guide** (Issue 4 is the latest version,
273 published in July 1992.)

274 The XSH internationalisation facilities represent the most comprehensive, commonly agreed
275 understanding of the requirement to date. They are summarised in Section 2.5.1.

276 Recent further work within the X/Open-Uniform Joint Internationalisation Group (JIG) has
277 been concerned with internationalisation within a distributed systems environment. This
278 concludes that the internationalisation facilities specified in the referenced **XPG4-XSH X/Open**
279 **CAE** specification are not sufficient. It proposes further facilities and places an implicit
280 requirement on the communication infrastructure. It represents the current direction of thinking
281 and is summarised in Section 2.5.2.

282 **2.5.1 XPG4 Facilities**

283 Firstly, the referenced **XPG4-XSH X/Open CAE** specification includes the `wchar_t` type of ISO C
284 and the locale mechanism of the referenced POSIX.1 international standard.

285 Secondly, recognising that many of the traditional open systems facilities do constrain the
286 language, culture or business environment assumed by the application, XSH includes a parallel
287 Worldwide Portability Interface facility for each such traditional facility. These facilities are
288 provided by the functions that are defined in the referenced ISO MSE addendum to ISO C.

289 While the referenced **XPG4-XSH X/Open CAE** specification includes both the traditional, non-
290 internationalised, function definitions and the internationalised, Worldwide Portability function
291 definitions, it recommends use of the latter for new developments, retaining the traditional
292 definitions for compatibility with existing systems and applications.

293 **2.5.2 Distributed Internationalisation Requirements**

294 The X/Open-Uniform Joint Internationalisation Group (JIG) has produced the referenced **DISS**
295 **Issue 1 X/Open** snapshot. This document discusses the issues arising from the need for
296 internationalised applications programs executing in a distributed environment. In particular,
297 when distributed internationalised applications cooperate, it is important that they assume the
298 same locale information. For example, if a list of names created on a system in Denmark is
299 sorted into alphabetical order on a system in the USA, the American system must use the right
300 collating rules (placing AA at the end of the list rather than at the beginning, for instance). For
301 this to be possible the following must be true:

- 302 • there must be a standardised means of describing locales
- 303 • there must be a way of identifying particular locales

- 304 • there must be a way of conveying locale information between communicating applications
305 • it must be possible for an application to use the appropriate locale when processing
306 information that has been created by another application.

307 The **DISS** contains a detailed proposal for a standardised way of describing locales, and
308 proposes that a registry of standard locales should be established. Methods of conveying locale
309 information between distributed applications are still being studied.

310 The DISS also proposes a set of functions for processing *self-announcing data*. Such data may
311 include indications of the locale or locales in which it should be processed. (The term *tagged data*
312 has also been used). The use of self announcing data would enable the applications to use the
313 appropriate locale or locales. This would support *multi-locale* applications that handle
314 information from several different locales at the same time.

315 2.6 Current Work

316 Work is continuing on the following topics.

317 2.6.1 Revision of the DISS

318 The **DISS** is being revised in the light of developments. The latest draft is significantly changed
 319 from the published snapshot. The set of functions defined has been changed substantially and
 320 provides consistent and comprehensive multi-locale support.

321 2.6.2 Definition and Registration of Locales

322 A registry of standard locales has been established by X/Open. The operation of the registry is
 323 described in the X/Open **Locale Registry** Procedures guide. The locales in the registry can be
 324 obtained from X/Open. At the time of writing, the registry contains some 20 locales, including
 325 Danish, Dutch, English (American and British), Faroese, German (Austrian, German and Swiss),
 326 Greenlandic, Hungarian, Icelandic, Italian, Japanese, Latvian, Lithuanian, Polish, Portuguese and
 327 Romanian locales.

328 2.6.3 Complex Text Languages

329 The locale mechanism currently defined in XPG4 covers the most commonly encountered
 330 differences between languages or cultural environments. However, it does not provide for all
 331 differences. In particular, it does not address the special needs of those languages that have been
 332 described as *complex text languages*. These can be defined as languages that have different
 333 layouts and forms of the text for presentation purposes and for processing purposes. These
 334 differences are generally concerned with:

- 335 • directionality
 336 For example, in Arabic, Farsi, Urdu, Hebrew and Yiddish, the text flows mainly from right to
 337 left but includes segments that must be read from left to right.
- 338 • shaping and composition of characters
 339 For example, in Arabic, each character has a different form depending on whether it stands
 340 alone, is at the beginning of a word, is in the middle of a word, or is at the end of a word.
- 341 • national numbers
 342 For example, in Arabic, Thai, Chinese and Bengali, there are numeric characters other than
 343 the normal Arabic numerals (Arabic uses Hindi numerals), and the encodings of the Arabic
 344 numerals (hex 30-39 in ASCII) should be understood as representing these characters rather
 345 than the Arabic ones when the text of these languages is processed.

346 The current state of work on complex text languages is embodied in the referenced **Layout**
 347 **Services** snapshot. This explains the difficulties associated with processing text written in these
 348 languages, and describes some facilities that could be added to the X/Open CAE to help
 349 overcome them:

- 350 • an opaque data structure (a layout object) that can be associated with a locale and that
 351 describes the characteristics of a piece of text written in a complex text language
- 352 • functions that:
 - 353 — manipulate layout objects
 - 354 and
 - 355 — transform text in accordance with the characteristics given in layout objects

- 356 • a new locale category (LO_LTYPE) which could be implemented as:
- 357 — an extended version of the LC_CTYPE category
- 358 or
- 359 — as part of the layout object data structure.

360 **2.6.4 Use of UNICODE/ISO 10646**

361 ISO 10646 represents a radical new direction in character set encoding standards. There are a
362 number of questions relating to its use that are not yet settled. These include:

- 363 • Should a standard locale, or perhaps several standard locales, that use the ISO 10646
364 encoding (or perhaps a related UTF encoding) be defined?
- 365 • Should all implementations support all characters defined in the referenced ISO 10646
366 international standard (that is, treat them as valid input and perform valid comparisons on
367 them), or should it be possible to define standard subsets so that an implementation need not
368 support every character?
- 369 • How should APIs (and particularly C language APIs) provide for character strings that may
370 be encoded in accordance with ISO 10646?
- 371 • Should ISO 10646, or perhaps a UTF, be specified as the standard encoding for use in certain
372 situations in Open Systems?
- 373 • Should UCS-dependent APIs (that is, APIs that assume that character data is encoded in
374 accordance with ISO 10646 UCS-2) be defined?

375 **2.6.5 Testing of Internationalised Components**

376 An internationalised system component should work in any language and cultural environment.
377 This means that it must be tested in conjunction with a number of locales. The question of what
378 locales should be used for testing purposes has been raised. It may be that new locales,
379 incorporating particular combinations of characteristics, will be defined for testing purposes.

380 **2.6.6 Distributed Internationalisation Framework**

381 A framework document that sets the context for work on internationalisation in distributed
382 systems is in preparation. It will provide an overview and analysis of the problem areas, but
383 will not contain detailed interface specifications, which will be in the DISS.

384 / *X/Open Technical Study*

385 **Part 2**

386 **X/Open Interworking Specifications**

387 *X/Open Company Ltd.*

Introduction

388

389 The chapters in this part of this technical study consider the impact of internationalisation on the
 390 X/Open interworking specifications. These chapters supersede the previously published
 391 X/Open **Interworking Internationalisation** snapshot.

392 The X/Open interworking specifications examined in this Technical Study are those that at the
 393 time of writing are already, or are expected shortly to become, CAE specifications. They consist
 394 of the following documents (full details are given in **Referenced Documents** on page xv).

- 395 • the **XTI** specification
- 396 • the **XMPTN** specifications (Access Node and Address Mapper)
- 397 • the **XAP**, **XAP-TP**, and **XAP-ROSE** specifications
- 398 • the **XOM** specification
- 399 • the **XFTAM** specification
- 400 • the **BSFT** specification
- 401 • the **X.400 API** specification
- 402 • the **XMS** specification
- 403 • the **XDS** specification
- 404 • the **XNFS** specification
- 405 • the **(PC)NFS** specification
- 406 • the **SMB Protocols** specification
- 407 • the **IPC Mechanisms for SMB**.

Structure of This Part

408 Chapter 4 describes the criteria that have been followed in identifying internationalisation issues
 409 in the X/Open interworking specifications.

411 Chapter 5 examines the implications of internationalisation on the interworking specifications
 412 listed above.

413 Chapter 6 presents conclusions and recommendations.

414 After this, Chapter 7 contains a set of *internationalisation* Change Requests (CRs) for the
 415 interworking specifications listed above. These are edited versions of standard X/Open Change
 416 Requests (CRs), in which the identity of the originator is omitted and the CRs are re-numbered
 417 into a sequential scheme, for the purposes of this document.

Potential Issues for Interworking

418

419 For the purposes of this technical study, occurrences in an X/Open interworking specification of
 420 either of the following requirements has been taken as a sign that there may be an
 421 internationalisation issue:

- 422 • A requirement for a particular character set or encoding (for example, ASCII) for textual
 423 information passed across an API or over a communications interface.

424 **Note:** A requirement for representation of non-textual information in a specific way has
 425 not been considered to give rise to internationalisation issues. For example, a date
 426 in ddmmyy form can be interpreted by a program operating in any language
 427 environment. It is only when that form is used at the user interface that there are
 428 problems. Similarly, a requirement for a constant text string expressed in a
 429 particular language (such as “LANMAN”) does not pose internationalisation
 430 problems since it is an arbitrary value that can be processed by programs as a
 431 constant.

- 432 • A requirement for any form of user interface.

433 Although it is possible to specify a user interface in an internationalised way, there are so
 434 many problems in this area that it is right to analyse any user interface requirement.

435 In considering the internationalisation issues that may arise from occurrences of the above
 436 requirements the following four criteria have been used.

437 Specification Portability

438 Can the specification be meaningfully implemented in any language and cultural
 439 environment?

440 For example, a requirement for a user interface which requires particular English language
 441 commands and responses does not meet this criterion. Nor does a requirement for an API
 442 in which the application must pass text encoded as 7-bit ASCII strings.

443 Applications Portability

444 Assuming portable specifications, can applications be written in such a way that they can
 445 operate under different language and cultural environments?

446 For example, a specification that requires text to be passed to the application *in the local*
 447 *language* can be implemented in any language environment but will not support
 448 internationally portable applications.

449 Implementation Interworking

450 Can implementations written in different language and cultural environments interwork?

451 For example, a specification that requires use of different national character set encodings
 452 derived from a single international superset would not necessarily produce
 453 implementations that interwork since there could be situations where a character code is
 454 legally usable in one country but not in another.

455 **Applications Interworking**

456 Can an application interwork with other applications or service implementations on remote
457 machines operating under different language and cultural environments?

458 For example, interworking could be a problem where one application sorts text generated
459 by another application on another system and the service does not enable the sorting
460 application to determine the collating sequence to be used. Note that this issue can arise
461 where two applications interwork using a service (for example, they communicate using
462 XTI) or where an application interworks with a service in a remote system (for example,
463 where an application uses XFTAM).

464 Although use of the POSIX Portable Filename Character Set is recommended in POSIX, it does
465 restrict users whose natural language is not English. For example, a Dane could not use his
466 spelling of the name which in English is spelt *Aarhus*. For this reason, although they may
467 provide applications portability, character set restrictions are identified in this technical study as
468 affecting specification portability.

470 5.1 The XTI Specification

471 5.1.1 Overview

472 The X/Open Transport Interface (XTI) Specification defines an API to a transport service that
 473 provides process-to-process communication and can be used in connection with OSI transport
 474 protocols, with TCP or UDP from the Internet Protocol Suite, with OSI Transport over TCP as
 475 defined in RFC 1006, with a minimal 7-layer OSI stack, with X.25, with SNA or with NETBIOS.⁴

476 5.1.2 Internationalisation Implications

477 The `t_error()` Function

478 The `t_error()` function defined in Chapter 6 of the XTI specification takes a null terminated
 479 character string as input, and writes to standard output an error message consisting of the input
 480 character string followed by a colon, a space, a standard error message describing the last
 481 encountered error, and a newline character. This raises a number of issues.

482 The input is a null terminated string. This is in accordance with the referenced ISO C
 483 international standard, but does not allow use of the referenced ISO 10646 international
 484 standard. This affects specification portability.

485 There is nothing to prevent, or even to discourage, the applications programmer from hard-
 486 coding the input strings into the program source code, rather than obtaining them from a
 487 catalogue associated with the currently established locale. This relates to applications
 488 portability (applications portability is not precluded, but nothing is done to encourage it).

489 A character set containing the newline, colon and space characters is assumed. These (in
 490 particular, the colon character) may not be meaningful in all language and cultural
 491 environments. This affects specification portability.

492 The input string is written to standard output followed by the standard error message. This may
 493 not be the most appropriate order in all language and cultural environments. This affects
 494 specification portability.

495 The standard error message strings are specified for the English language but are
 496 implementation-defined for other languages. There is nothing, however, to say that the
 497 standard error message string that is output should depend on the currently established locale.
 498 This impacts on applications portability; it will not be possible to write an internationalised
 499 application that is portable between implementations that assume different natural languages.
 500 For example, an application running on an English implementation would display English
 501 messages, even to French users.

502

503 4. The appendices dealing with the minimal 7-layer OSI stack, with X.25 and with SNA are yet to be published. Drafts have been
 504 reviewed for this technical study. They contain no internationalisation implications.

505 A change request to make the error message depend on the current locale (see Change Request
506 XOP-1 in Chapter 7) has been accepted, and will be implemented in the next published version
507 of the specification. A further change request to replace type **char** by type **wchar_t** in the *errmsg*
508 argument of *t_error()* was not accepted. However, X/Open is to consider specifying a parallel
509 function that will use **wchar_t** rather than **char**.

510 **The *t_strerror()* Function**

511 In the description of *t_strerror()* in Chapter 6 of the XTI specification, it is stated that a string is
512 returned.

513 The string returned is null terminated. This is in accordance with the referenced ISO C
514 international standard, but does not allow use of UNICODE or ISO 10646. This affects
515 specification portability.

516 Specific text is provided for the case where the natural language is English and it is stated that
517 “in other languages, an equivalent text is provided.” This impacts on applications portability
518 (as in the case of the string generated by *t_error()*, described above).

519 As for *t_error()*, a change request to make the error message depend on the current locale (see
520 Change Request XOP-1 in Chapter 7) has been accepted, and will be implemented in the next
521 published version of the specification, but a further change request to replace type **char** by type
522 **wchar_t** in the *errmsg* argument of *t_error()* was not accepted.

523 **NetBIOS**

524 In section D.5 of the XTI specification, the description of NetBIOS names prohibits the first octet
525 from being null or hexadecimal FF, prohibits the last octet from being in the range (hexadecimal)
526 00-1F, and reserves special meaning to the ASCII code for the asterisk character. This restricts
527 the codesets that may be used, and hence affects specification portability.

528 This problem derives from the NetBIOS specification, rather than the XTI specification. It is not
529 appropriate to address it by changing XTI.

530 5.2 The XMPTN Specification

531 5.2.1 Overview

532 The X/Open Multi-Protocol Transport Networking (XMPTN) architecture supports mixed
533 protocol networking. It enables an application that was designed to run over a single, specific
534 protocol (such as SNA, NetBIOS, OSI Transport or TCP/IP) to run over additional networks
535 using different protocols.

536 The XMPTN architecture includes:

- 537 • *Access Nodes*
- 538 • *Address Mappers.*

539 In an XMPTN *access node*, an application program that uses communications transport services
540 (a *transport user*) interfaces, through XMPTN components, to a *transport provider* that provides
541 those services. The transport provider may implement a communications protocol other than
542 the one whose use is assumed by the transport user. The XMPTN components translate the
543 transport service requests made by the user into the transport service primitives provided by the
544 transport provider. The XMPTN components also provide compensation for services assumed
545 by the transport user but not implemented by the transport provider, by implementing those
546 services using the services that the transport provider does provide. The components of an
547 XMPTN access node, and the way they operate with transport users that use TCP/IP, UDP/IP,
548 SNA, NetBEUI and NetBIOS, are described in the referenced **XMPTN Access Node X/Open**
549 preliminary specification.

550 An XMPTN *address mapper* holds details of the relationships between transport addresses used
551 by transport users and transport addresses used by transport providers. An access node can
552 communicate with an address mapper, using a transport service, to register address mappings
553 and to determine the transport provider addresses corresponding to a transport user address.
554 The function of an XMPTN address mapper is described in the referenced **XMPTN Address**
555 **Mapper X/Open** preliminary specification.

556 5.2.2 Internationalisation Implications

557 There may be internationalisation implications in the address formats of some of the transport
558 protocols mapped by XMPTN. However, the purpose of XMPTN is to allow existing protocols
559 to continue to be used in a multi-protocol environment, rather than to provide a new transport
560 mechanism. In general, these implications should therefore be recognised as limitations
561 deriving from the use of the existing protocols, rather than as defects in XMPTN that should be
562 corrected by changing XMPTN.

563 A slightly different case is that of TCP transport providers, for which transport user addresses
564 may be resolved by being converted to ASCII host names and looked up using the Internet
565 Domain Name Service. It is not explained how names expressed in codesets that do not readily
566 convert to ASCII should be handled. It would be possible to derive coding schemes, such as that
567 used for the algorithmic mapping used to convert IP addresses to SNA LU names, in which any
568 address encoding can be converted to ASCII. Unless such schemes are used, there will be
569 specification portability issues arising from the use of XMPTN with TCP transport providers.

570 **5.3 The XAP, XAP-TP and XAP-ROSE Specifications**

571 **5.3.1 Overview**

572 The X/Open ACSE/Presentation Services API defines an Application Program Interface to the
573 Association Control Service Element (ACSE) of the OSI Applications Layer and the OSI
574 Presentation Layer Service, excluding the encoding of information in ASN.1 (this is left as the
575 responsibility of the application). It is not based on the XOM API (which provides an interface
576 to ASN.1), although it can be used in conjunction with XOM.

577 The XAP-ROSE API is an extension of the XAP API that provides an interface to the OSI Remote
578 Operation Service Elements (ROSE).

579 The XAP-TP API is an extension of the XAP API that provides access to the services of the OSI
580 Transaction Processing (TP) protocol.

581 **5.3.2 Internationalisation Implications**

582 **Service Provider Identifiers**

583 Service providers are identified by null-terminated strings. This would cause problems for an
584 application using the ISO 10646 codeset, and hence affects specification portability.

585 **Diagnostic Messages**

586 The *ap_get_env()* function returns a diagnostic message in field *error* of structure *ap_diag_t* when
587 passed the AP_DIAGNOSTIC attribute. This message is in the natural language of the currently
588 defined locale. However, it is contained in a null-terminated character string. There is therefore
589 a question of specification portability, since null-terminated character strings can not be used in
590 conjunction with the codeset of ISO 10646. X/Open is to consider specifying a parallel function
591 that will use **wchar_t** rather than **char**.

592 **Error Messages**

593 The *ap_error()* function returns a pointer to an error message in the language of the currently
594 specified locale. Again, there is a specification-portability issue, because the message is
595 contained in a null-terminated string, and X/Open is to consider specifying a parallel function
596 that will use **wchar_t** rather than **char**.

597 **XAP-ROSE**

598 There are no internationalisation implications for the referenced **XAP-ROSE** X/Open
599 preliminary specification.

600 **XAP-TP**

601 There are no internationalisation implications for the referenced **XAP-TP** X/Open preliminary
602 specification.

603 5.4 The XOM Specification

604 5.4.1 Overview

605 The X/Open Abstract Data Manipulation (**XOM**) specification defines a general purpose OSI
606 Abstract Data Manipulation Application Program Interface (API) for use in conjunction with
607 other X/Open application specific APIs for Open Systems Interconnection (OSI). XOM deals
608 with *information objects* that arise in OSI, ie. those that can be expressed in terms of ASN.1 The
609 specification defines how objects can be created, examined, modified and deleted.

610 Character String Types

611 The definition of ASN.1 in the referenced ISO 8824 international standard (the version on which
612 the XOM was originally based) specifies the characters that can be represented in the various
613 sorts of string either by explicitly defining them or by referring to character set registration
614 numbers in ISO 2375. The types of string defined in the referenced ISO 8824 international
615 standard are as follows:

- 616 • *General String* - all internationally registered graphic and control character sets (plus SPACE
617 and DELETE)
- 618 • *Graphic String* - all internationally registered graphic character sets (plus SPACE)
- 619 • *IA5 String* , *VisibleString (ISO646String)*, *PrintableString* - variations of the basic ASCII
620 character set
- 621 • *TeletexString (T61String)*, *VideotexString* - certain identified internationally registered graphic
622 and control character sets.

623 Since that version, the definition of ASN.1 has been revised. Three new types of character string
624 have been added. They are:

- 625 • *Universal Strings* - strings of characters encoded according to ISO 10646 UCS-4 form
- 626 • *BMP Strings* - strings of characters encoded according to ISO 10646 UCS-2 form
- 627 • *Unrestricted Strings* - strings of characters with a syntax that has an ASN.1 object identifier
628 and which either
 - 629 — are type-compatible with characters of one of the old character string types or the new
630 Universal String type,
 - 631 or
 - 632 — have a possible transfer syntax (that is, an encoding) that has an ASN.1 object identifier.

633 The Unrestricted String type allows use of any character set and encoding once the necessary
634 object identifiers have been allocated to them. In particular, it allows use of the character sets
635 and encodings defined in ISO 2375 and ISO 10646.

636 The string type definitions in XOM reference the original ASN.1 character string definitions, and
637 also the new Universal and Unrestricted string types, but not the new BMP String type.

638 5.4.2 Internationalisation Implications

639 The referenced **XOM X/Open CAE** specification does not allow for the ASN.1 BMPString type
640 introduced in the ISO/IEC 8824:1 1994 standard. The addition to XOM of a BMP character string
641 syntax corresponding to this ASN.1 type should be considered.

642 While the string types defined in XOM include types that are capable of representing any
643 national character set, they also include types that are restricted to representing only some
644 national character sets. The use of these string types in APIs that use XOM thus requires careful
645 consideration to ensure that those APIs are fully “internationalised”. The possible use in these
646 APIs of string types corresponding to the new Universal and Unrestricted String types
647 consideration. These aspects are discussed in Section 5.7 on page 32 and Section 5.9 on page 34
648 of this technical study.

649 5.5 The XFTAM Specification

650 5.5.1 Overview

651 The XFTAM specification defines an API to the OSI File Transfer, Access and Management
652 (FTAM) service. It supports file types FTAM-1, FTAM-2 and FTAM-3. It is defined using the
653 X/Open OSI Abstract Data Manipulation Service (XOM).

654 5.5.2 Internationalisation Implications

655 Identifier Strings

656 File names, creator identities etc. are represented in the API by XOM Graphic Strings. This
657 means that any internationally registered character set can be used. It does not allow use of
658 UNICODE/ISO 10646, however, or have the full generality of the Unrestricted Strings of the
659 referenced DIS 8824 draft international standard. There is therefore an impact on specification
660 portability.

661 The XFTAM API reflects the referenced ISO 8571 international standard and should not be
662 changed independently of it. Work on the FTAM standards should be kept under review, and
663 any change in them to permit use of Universal or Unrestricted Strings should be reflected in
664 XFTAM. At the time of writing, this issue has been raised in ISO, but there is no formal defect
665 report, and no concrete plan to amend the FTAM standards.

666 File Contents

667 The FTAM-1 and FTAM-2 document types specify text files. Conversion of format-effector
668 characters (in particular, *end of line*) is specified to occur on input and output. All of the XOM
669 character string types except Universal String and Unrestricted String are supported. As noted
670 for Identifier Strings above, this is not completely general, and there is an impact on specification
671 portability. No change should however be made to XFTAM until a corresponding change has
672 been made to the referenced ISP 10607-2 international standardised profile, in which the
673 document types are defined. As with the types of file names etc. (discussed above), there is no
674 concrete work plan to address this issue.

675 Although the application using XFTAM will in general be aware of the codeset used in a text file
676 (from the Content-Class OM attribute of the Content-Type OM attribute of the FTAM-Attributes
677 objects returned by the interface functions), it will not be aware of the collating sequences, case
678 conversion rules etc. that apply. There is thus an implication for applications interworking. To
679 address this, it would be necessary for locale information to be associated with the file in some
680 way, and to be passed to the remote application. Whether this is desirable and, if so, how it
681 could be achieved, is for further study.

682 Diagnostic Text

683 The FTAM-Output-Parameters OM class, which defines a class of OM object that is returned by
684 several of the API functions, contains OM attribute FTAM-Diagnostic-List, which is of syntax
685 Object(FTAM-Diagnostic), and in turn contains an OM attribute, Text-Message, which is of
686 syntax String(Graphic) and contains an optional text message in natural language.

687 For applications interworking, this message should be in the natural language of the current
688 locale. However, since the text is generated in the remote system (the FTAM responder), this
689 would imply some means of conveying locale information between initiator and responder.
690 This is an issue for further study.

691 5.6 The BSFT Specification

692 5.6.1 Overview

693 The X/Open **BSFT** Specification defines a file transfer utility similar to the IPS ftp utility but
694 using the OSI FTAM protocol. It includes a specification of a protocol profile and a user
695 interface.

696 5.6.2 Internationalisation Implications

697 User Interface

698 The user interface (defined in **BSFT** Appendix A) is specified in English. This clearly impacts on
699 Specification Portability.

700 The command and argument names use only the characters of the portable filename character
701 set of the referenced ISO 9945-1 international standard and the interface should therefore be
702 usable wherever the character set is derived from the Latin alphabet, although the command and
703 argument names will not be meaningful in languages other than English.

704 The user interface should be defined in such a way that a meaningful version of it can be derived
705 for any natural language environment. This could be done by requiring that standard
706 Internationalisation facilities are used for display of dates and times and for collating file names
707 and that command and argument names are held in natural language specific message
708 catalogues that can be referenced through locale information.

709 File Types

710 BSFT supports both text files (type FTAM-1) and binary files (type FTAM-3). There are no
711 internationalisation issues relating to the contents of binary files, but there are two implications
712 relating to text files.

713 First, as for XFTAM, the contents of text files can be of various types, including Visible String,
714 IA5 String, Graphic String or General String, but this is not sufficiently general to cater for all
715 possible character sets and encodings, and there is thus an implication for specification
716 portability.

717 Secondly, there is a possible impact on implementation interworking relating to text files. BSFT
718 can be used to transfer text files between two locales that use the same codeset (or that use
719 codesets that are closely related) but will not effect a meaningful transfer of textual information
720 between locales that use radically different codesets (for example, Hebrew and Korean). Such a
721 transfer is clearly beyond the scope of BSFT. However, its behaviour if such a transfer is
722 attempted is currently not defined. It should ideally be defined in a fully internationalised
723 specification. (Note that this problem does not arise in relation to the **XFTAM** specification,
724 since the codeset used is known to an XFTAM application from the Content-Class OM attribute,
725 as discussed under **File Contents** on page 29 of this technical study.) This issue is not specific to
726 interworking; the same problem can arise when transferring information between files created
727 using different locales on a single machine. It raises the question of whether locale information
728 could or should be associated with files in some way. In an interworking context, it also raises
729 the question of how the locale information could be conveyed between the communicating
730 systems.

731 **Names**

732 Filenames, user identities, account names and passwords can be of type Graphic String, and
733 wildcard expansion can be requested. Again, there is an impact on specification portability, in
734 that the possible codesets used are restricted, and there is an impact on implementation
735 interworking, in that this allows use of BSFT between two locales with similar character sets but
736 not between locales with different character sets. Again, such use is beyond the scope of BSFT
737 but the behaviour of BSFT when such use is attempted should be specified.

738 **Natural Language Text in PDUs**

739 Protocol error messages can contain details expressed in natural language and Initialise
740 Request/Response PDUs may include implementation information expressed in natural
741 language. This has a possible impact on implementation interworking. If these textual
742 messages can be displayed at the user interface (**BSFT** Appendix A does not specify either that
743 they should be or that they should not be) then similar considerations apply as for filenames,
744 user identities etc. In addition, even where such information is transferred between two
745 communicating systems that use similar codesets, the user may not understand it since it may
746 not be expressed in his natural language.

747 5.7 The X.400 API Specification

748 5.7.1 Overview

749 The X/Open X.400 API Specification defines:

- 750 • an X.400 Application API that makes the functionality of a message transfer system (MTS)
751 accessible to a message store (MS) or user agent (UA)
- 752 • an X.400 Gateway API that divides a message transfer agent (MTA) into two software
753 components: a mail system gateway and an X.400 gateway service.

754 It is defined using the X/Open OSI-Abstract-Data Manipulation service (XOM).

755 5.7.2 Internationalisation Implications

756 The API is based on the definitions of the referenced X.400 CCITT Recommendations which are
757 essentially the same as the referenced ISO 10021 international standard. They can therefore be
758 expected to be fully *international*. In fact, the only internationalisation implications of the API
759 specification derive directly from the CCITT X.400 Series recommendations. This is the fact that
760 certain class attributes are defined to be Teletex Strings, Videotex Strings or Printable Strings
761 which (as discussed in the section on XOM) restricts them to certain, internationally registered,
762 character sets. The permitted character sets include all Western European characters and
763 Japanese Kanji (set nr. 87 in the ISO register) but not other sets such as Hebrew and Arabic.

764 In some cases (for example, that of the **Teletex Document** attribute of the **Teletex Body Part**
765 class), this restriction is inevitable and can not be considered to impact on internationalisation. It
766 makes no sense to use a non-teletex character set in a Teletex Body Part. In other cases, notably
767 those of many of the attributes of class *OR-Address*, the restriction is harder to understand.

768 The definitions meet the needs of Western Europe, North America and Japan but not of other
769 countries. It is not clear why the X.400 recommendations do not specify Graphic Strings,
770 General Strings or even Octet Strings, instead of Teletex Strings. This would enable characters of
771 any internationally registered character set to be printed. It is possible that future versions could
772 specify the Universal, BMP or Unrestricted Strings of the new DIS 8824.

773 There is thus an impact on specification portability deriving not from the API specification but
774 from the underlying X.400 series recommendations of the CCITT.

775 Changes should not be made in advance of changes to ISO 10021 and the X.400 series
776 recommendations. The progress of these standards should be kept under review and the **X.400**
777 **API** Specification should be modified to reflect any changes that are made to them to address
778 this issue.

779 ISO SC18 WG4 has had an outstanding work item to address this issue for the last three years.
780 However, no national body has yet made a concrete proposal. ISO SC18 WG4 would welcome
781 input from X/Open on the appropriate timescale for addressing the issue.

782 5.8 The XMS Specification

783 5.8.1 Overview

784 The X/Open XMS API provides an API to message store functions similar to those described in
785 X.413 (see the referenced X.400 CCITT Recommendations). It is complementary to the X.400 API
786 and, like the X.400 API, it is defined using the X/Open OSI-Abstract-Data Manipulation service
787 (XOM). Also, it uses some of the class definitions from the X.400 and XDS APIs.

788 5.8.2 Internationalisation Implications

789 As with the X.400 API, there are a number of attributes that represent text strings and that are
790 restricted by their syntaxes to certain character sets. These include the following:

- 791 • the **Content-Identifier** OM attribute of OM class **Auto-Forward-Arguments**, which has
792 syntax **Printable String**
- 793 • the **IA5-String** OM attribute of OM class **Password**, which has syntax **IA5 String**
- 794 • the **A-Content-Identifier** MS attribute, which has syntax **Printable String**
- 795 • the **IM-Auto-Forward-Comment** IM attribute, which has syntax **Printable String**
- 796 • the **IM-Languages** IM attribute, which has syntax **Printable String**
- 797 • the **IM-Subject** IM attribute, which has syntax **Teletex String**
- 798 • the **IM-Suppl-Receipt-Info** IM attribute, which has syntax **Printable String**
- 799 • the **Subject** OM attribute of OM class **Heading**, which has syntax **Teletex String**.

800 As with the X.400 API, there is an impact on specification portability which derives from the
801 underlying X.400 series recommendations of the CCITT, rather than from the API specification.

802 Changes should not be made in advance of changes to ISO 10021 and the X.400 series
803 recommendations. The progress of these standards should be kept under review and the **XMS**
804 API specification should be modified to reflect any changes that are made to them to address
805 this issue.

806 5.9 The XDS Specification

807 5.9.1 Overview

808 The X/Open API to Directory Services (XDS) defines an API to directory services that include,
809 but are not limited to, those defined in the referenced X.500 CCITT Recommendations. The
810 assumed model of a Directory, and many of the associated definitions, are directly derived from
811 those CCITT recommendations (which are aligned with and technically equivalent to the
812 referenced ISO 9594 international standard).

813 The API is defined using the X/Open OSI-Abstract-Data Manipulation service (XOM).

814 5.9.2 Internationalisation Implications

815 Attribute Character Sets

816 For certain attributes - notably **A-Country-Name** and **A-Destination-Indicator** - there appears to
817 be an issue because they are defined as (Latin alphabet) printable strings. This is necessary,
818 however, in order to allow addresses to be recognisable internationally. (The **A-Country-Name**
819 attribute, for example, is used to hold the standard country codes defined in ISO 3166). No
820 action should therefore be taken to modify the XDS in respect of these attributes.

821 Attribute Comparisons

822 In the definitions of OM classes **Filter-Item** and **Search-Criterion**, various attribute comparison
823 methods are discussed.

- 824 • Attribute matches can be “approximate” using an implementation-dependent algorithm. It
825 is probable that full *internationalisation* requires the algorithm to take account of the locale of
826 the user and/or the subject of the directory entry - approximate matching may well be
827 different in England and Japan. There is thus again an impact on specification portability.

828 Consideration could be given to adding a *locale* OM attribute to OM class **filter** (to allow the
829 user’s locale to be specified) and to providing a mechanism for the algorithm to take account
830 of any *locale* (directory) attribute in the directory entry. However, it should be noted that the
831 definition of **filter** is based closely on definitions in CCITT Recommendation X.511. It should
832 not be changed by X/Open until corresponding changes have been made to the Directory
833 Standards by ISO and the CCITT.

- 834 • Greater-or-equal/less-or-equal comparisons are made using “the appropriate ordering
835 algorithm”. For strings, it is probable that this algorithm should use collating sequence tables
836 for the locale of the user and/or the entry and should cater for “right-to-left” (eg. Arabic) as
837 well as “left-to-right” text strings. Again, these rules are defined in the CCITT X.500 series
838 recommendations and X/Open should not change the XDS until corresponding changes
839 have been made to the Directory Standards by ISO and the CCITT.

840 The 1993 version of the Directory Standards includes provision for strings of ASN.1 type
841 Universal String (but not for strings of type BMP String or Unrestricted String). Since Universal
842 String maps to the 4-octet ISO 10646 representation rather than to the 2-octet representation
843 (which is probably the more commonly used), there is a proposal to add support for BMP String,
844 which does map directly to the 2-octet representation. There is currently no intention of adding
845 support for Unrestricted character strings.

846 There is a further change proposed to the Directory Standards that could enable all the
847 outstanding internationalisation issues to be resolved. This is that all directory attributes should
848 be allowed to have a number of properties. The properties that have been considered include
849 *language* or *locale* (*language*, rather than *locale*, seems to be favoured currently). (Other properties,
850 such as the *time* when the validity of the attribute expires, are being considered also.)

851 The progress of these proposals should be monitored, and the XDS should be modified to reflect
852 any resulting changes in the standards. It would indeed be possible to modify XDS now to take
853 account of the new Universal String type. However, the decision on what changes to make, and
854 when to make them, should take into account:

- 855 • the possibility that the Directory standards will be modified to cater for the BMP String type
- 856 • the other possible changes to the Directory standards discussed above
- 857 • the fact that further changes (not related to internationalisation) have been made to the
858 Directory standards, and that it may be desirable to modify the XDS to reflect them.

859 5.10 The XNFS Specification

860 5.10.1 Overview

861 With the XNFS specification, X/Open provides a temporary but complete solution to the
862 problem of transparent file access between X/Open-compliant systems. XNFS is described as
863 temporary because X/Open recognises that the Transparent File Access (TFA) standardisation
864 work is ongoing within the IEEE P1003.1f project, and X/Open intends to be compliant with
865 P1003.1f TFA when it becomes an IEEE standard. XNFS is considered complete because it
866 encompasses both protocols for interoperability (via the XNFS specification) and interfaces for
867 application/user portability (via the XSI specification and the semantic differences defined in
868 the appendices of the XNFS specification).

869 XNFS comprises a number of specifications, namely:

870 External Data Representation (XDR)

871 This defines a syntax for describing data formats and data encoding (analogous but not
872 equivalent to ASN.1). XDR is used to specify the other XNFS protocols.

873 Remote Procedure Call Protocol (RPC)

874 This provides a mechanism to allow a client to call a procedure to be executed on a remote
875 server.

876 Network File System (NFS)

877 The X/Open specification for file-sharing services based on the NFS architecture developed
878 by Sun Microsystems Inc.

879 Portmap

880 A service that maps RPC program and version numbers to transport-specific port numbers
881 thus providing a dynamic binding capability for remote programs.

882 Mount

883 A service that looks up server pathnames, validates user identities and checks access
884 permissions. It then provides the first file handle to clients, which then allows them entry to
885 a remote file system.

886 Network Status Monitor (NSM)

887 A service that provides applications with information on the status of network hosts. It is
888 used by NLM to track hosts that have established and hold locks.

889 Network Lock Manager (NLM)

890 An RPC-based service that provides advisory X/Open CAE file and record locking and
891 DOS compatible file sharing and locking in an XNFS environment.

892 5.10.2 Internationalisation Implications

893 There are two types of potential Internationalisation implication in XNFS: those that are protocol
894 related and those that arise out of Transparent File Access (TFA). The former correspond to the
895 issues of Specification Portability and Implementation Interworking and the latter to the issues
896 of Application Portability and Interworking.

897 **Protocol Issues**

898 The point at issue here is whether the encoding of parameters is language-dependent. The XNFS
899 set of protocols uses a variety of parameter types that are clearly language-independent (for
900 example, boolean and integers) but it also uses *string* parameters. Strings are used to specify
901 program names, path names, and user names. However, the specification treats these as octets
902 and does not process them, other than to transmit, store, retrieve and compare them for equality.
903 Consequently, as far as the protocol implementation is concerned, string parameters are
904 language-independent, and protocol implementations are both portable and, when implemented
905 on systems with different cultural environment, will interwork. The only proviso is that strings
906 consist of 8-bit octets.

907 Where the XNFS protocols (such as RPC) are used by other programs, then the unrestricted use
908 of RPC protocols may give rise to internationalisation problems. These problems cannot be
909 easily solved through the use of the current internationalisation features of the X/Open CAE
910 since the code will be running on one machine (and have access to the machine's language
911 dependent features) but will have to code user names, path names etc. using the language-
912 dependent features of another machine. These issues are being addressed by the Joint X/Open-
913 Uniforum Internationalisation Group, as discussed in Section 2.5.2 on page 12 of this technical
914 study.

915 **TFA Problems**

916 The purpose of XNFS is to provide transparent file access. This gives rise to issues that are,
917 potentially, much more serious than those associated with the XNFS protocols. These issues are
918 identified in the referenced **DISS Issue 1** X/Open snapshot; they derive from the fact that client
919 and server do not necessarily have a common locale.

920 There is a similar problem with the FTAM initiator and responder for XFTAM and BSFT, but the
921 situation is worse for XNFS. With FTAM, the codeset in use is known to both the initiator and
922 the responder, it is only case conversion rules, collating sequences etc. that are not known, and
923 these are only required by a minority of applications. With XNFS, not even the codeset is
924 known.

925 The following example illustrates the problem. Suppose that a French user creates a file *données*
926 on his system, which supports the codeset of ISO 8859-1. His system is networked to an English
927 system that does not support this codeset, but simply ignores the most significant bit of any
928 character code. The two systems are running XNFS. A directory listing of the French system,
929 requested on the English one, displays the filename as *donnies*, the code (hex) E9 for the letter e-
930 with-acute having been displayed as though it were (hex) 69. However, attempts to access the
931 file *donnies* are unsuccessful, because the English system transmits code (hex) 69 to the French
932 one, rather than code (hex) E9.

933 This problem is pointed out in the XNFS Specification. It can be addressed by enabling the user
934 to establish identical locales on all the systems that he uses, and by ensuring that the same locale
935 is used by all co-operating applications in a distributed operation. The user in the above
936 example could then establish a French locale on the English system and work in it, using files
937 located on his French system, as though he were working on his French system.

938 The establishment of identical locales on systems supplied by different vendors implies that
939 locales must be standardised. Use of the same locale by co-operating distributed systems
940 implies that there must be some means of conveying locale information between those systems.

941 A further issue is that a user application may not be aware of the locale associated with a remote
942 file. This issue is not confined to distributed operations, as the same situation may arise with an
943 application accessing a local file, but it is more likely to arise in a distributed system. It raises the
944 question of whether locale information could, or should, be associated with files.

945 5.11 The (PC)NFS Specification

946 5.11.1 Overview

947 The X/Open (PC)NFS specification covers one of the two protocols sets that X/Open defines in
948 order to provide interoperability over Local Area Networks (LANs) between personal
949 computers and X/Open-compliant systems. (The other protocol, SMB, is discussed in Section
950 5.12 on page 40 of this technical study).

951 The (PC)NFS specification covers all of the protocols specified in the XNFS specification with
952 the following exceptions:

- 953 • the **Network Status Monitor (NSM)** service is not used
- 954 • the **Network Lock Manager (NLM)** service is extended to provide additional services
- 955 • a new service, **PCNFSD**, is used.

956 With the exceptions listed above, (PC)NFS reproduces, with editorial changes, the XNFS
957 specifications rather than referencing them.

958 The added facilities are as follows:

959 **Network Lock Manager (NLM)**

960 This is an RPC-based service that provides advisory X/Open CAE file and record locking,
961 and DOS-compatible file sharing and locking in an XNFS environment. The (PC)NFS
962 definition of NLM defines an upwardly compatible NLM specification, which is defined to
963 include all the facilities of the XNFS definition and adds to it support for personal
964 computers (namely non-monitored locks and DOS-compatible file sharing). The extra calls
965 include parameters that specify file caller names.

966 **Personal Computer NFS Daemon (PCNFSD)**

967 This is a Unix daemon that provides user authentication and print services to single-user
968 personal computer systems. (PC)NFS includes parameters to define passwords, host
969 names, printer names, user names, filenames and spool options. The specification includes
970 the definition of specific English characters for spooler options. However, since these are
971 fixed (and hence can be treated by programs as arbitrary constants), this is not an
972 internationalisation issue.

973 5.11.2 Internationalisation Implications

974 The discussion with regard to internationalisation of the XNFS protocol specifications also
975 applies to the equivalent (PC)NFS versions, with the extra complication that PCs handle national
976 language-dependent features in a different manner to X/Open compliant systems. In particular,
977 different encodings of characters will be used between PCs and X/Open compliant systems, and
978 PCs can change their character sets (code pages) without this becoming known to the X/Open
979 compliant systems to which the PC is connected.

980 Strings, such as machine names, program identifiers and procedure identifiers will be expressed
981 by PCs according to the DOS code page they are currently using. This will require 8-bit
982 character support and a character encoded in the same manner may be displayed differently to
983 users on Unix and PC systems.

984 The same sort of problems will arise as were illustrated in the example in Section 5.10 on page
985 36, but will be less easy to solve because the code page mechanism used by the PC clients is
986 different from the locale mechanism used on the X/Open compliant servers and the two
987 mechanisms may (for example) use different codesets for the same natural language.

988 5.12 The SMB Protocols Specification

989 5.12.1 Overview

990 SMB is a protocol that can be implemented as an *upper layer* over any protocol that provides the
991 NetBIOS service. The X/Open **SMB** specification describes the SMB protocol itself and its use of
992 NetBIOS. It also describes an OSI protocol stack and an IPS protocol stack that provide the
993 NetBIOS service (it does this by reproducing a MAP/TOP Users' Group Technical Report and
994 RFCs 1001 and 1002).

995 5.12.2 Internationalisation Implications

996 System Names, Resource Names and Passwords

997 SMB host names are upper-case characters padded with blanks. SMB names for resources (files,
998 paths, mailslots, pipes, volumes, devices, etc.) and also passwords are encoded as null
999 terminated ASCII strings. Pathnames may or may not be case-sensitive, and case conversion of
1000 filenames and path names is performed under some circumstances. The character set and
1001 encoding used is assumed to be ASCII and certain encodings — those for asterisk, question mark
1002 and back-slash and other special characters, and those less than (hex) 20 — have reserved
1003 meanings. Encodings greater than or equal to (hex) 80 need not be supported, and case
1004 conversion does not apply to them in any case.

1005 This clearly impacts on specification portability. It would for example be difficult in some
1006 language and cultural environments to write an electronic mail application in which a user could
1007 freely specify mailbox names.

1008 Some of the language and cultural dependencies could be removed by allowing any character
1009 encoding scheme that is an extension of ASCII. This would, however, still leave case conversion
1010 as a problem, and would not allow use of UNICODE/ISO 10646.

1011 On PCs, the algorithm for case conversion is likely to be determined by the code page in use. For
1012 case conversion to be meaningful with an extension of the ASCII encoding scheme, the X/Open
1013 compliant servers would have to be aware of the code pages in use on the PC clients. This
1014 would require an extension to the SMB protocol. The X/Open compliant servers would also
1015 need knowledge of the individual code pages used on the PCs. How this could be achieved
1016 requires further study.

1017 Data

1018 SMB is capable of carrying 8-bit binary data. However, in section 9.1, SMBsplopen has
1019 smb_mode which can specify text mode, allowing the server to expand ASCII tabs to spaces.
1020 This options could have a slight impact on applications portability. If writing an internationally
1021 portable application that used SMBsplopen to create a spool file, the programmer would have to
1022 be careful to use graphics rather than text mode. This represents a restriction on the range of
1023 tools available to the programmer, however, rather than a restriction on what can be achieved
1024 using those tools.

1025 **Management Transactions**

1026 In section B.4.1, the parameter descriptor string, the data descriptor string and the auxiliary data
1027 descriptor are null-terminated ASCII strings. These follow a particular coded format and so are
1028 *international*. However, the type of argument passed can include a null-terminated ASCII string
1029 but not other types of character string.

1030 This impacts slightly on applications portability. If writing an internationally portable
1031 management application, the programmer must be careful to avoid using parameters that are
1032 null-terminated ASCII strings. Again, the programmer is not restricted in what can be achieved,
1033 since it is possible to encode text in parameters in other ways (and, for an internationally
1034 portable application, it may be better in any case to avoid the use of text altogether).

1035 **5.13 The IPC Mechanisms for SMB Specification**

1036 **5.13.1 Overview**

1037 This X/Open specification defines an API to the Server Message Block (SMB) protocol for use on
1038 X/Open compliant systems, provides information on the mapping of the API to SMB protocol
1039 elements and specifies the protocol elements that are required.

1040 **5.13.2 Internationalisation Implications**

1041 As regards the data passed over the API and transported using the SMB protocol, there are no
1042 limitations that affect internationalisation. The names of resources (mailslots, pipes etc.) are
1043 however required to be null-terminated ASCII strings and some comment strings transferred by
1044 the protocol are also null-terminated ASCII strings.

1045 This clearly impacts on specification portability, as discussed in Section 5.12 on page 40.

Conclusions and Recommendations

1046

1047 This chapter summarises the implications of internationalisation requirements on X/Open
1048 interworking specifications and presents conclusions and recommendations.

1049 From the point of view of internationalisation, applications programs can be divided into three
1050 classes:

- 1051 • an application that requires a single, fixed locale can be classed as a *single-locale* application
- 1052 • an application can be classed as a *variable-locale* application if, at any time, it only processes
1053 data from a single locale, but that locale can vary from time to time
- 1054 • an application that processes data from several different locales at the same time can be
1055 classed as a *multi-locale* application.

1056 For single-locale applications, the only question is whether there is any restriction on the locales
1057 in which it can be implemented. Instances where an X/Open interworking specification
1058 imposes a restriction, or allows an implementation to impose a restriction, are identified in this
1059 technical study as issues of specification portability. Except for some issues arising from the use
1060 of specifications that X/Open can not change unilaterally, the only restriction now imposed by
1061 X/Open interworking specifications is that some API functions can not be used in conjunction
1062 with UNICODE or ISO 10646 codesets, because the functions rely on strings being null-
1063 terminated.

1064 The question of UNICODE/ISO 10646 has been discussed by X/Open working groups. There
1065 has been a reluctance to replace function arguments of type **char** with arguments of type
1066 **wchar_t**. However, X/Open will now consider defining additional functions that perform the
1067 same actions as the API functions that rely on strings being null-terminated, but that use arrays
1068 of type **wchar_t** rather than arrays of type **char**. These functions should not replace the existing
1069 functions, but should provide an alternative to them for use by internationalised applications.
1070 The functions affected are the XTI functions *t_error()* and *t_strerror()*, and the XAP functions
1071 *ap_get_env()* (which can return an **ap_diag_t** value) and *ap_error()*.

1072 For variable-locale applications, there are further questions that arise. These are:

- 1073 • whether the specification requires the implementation to allow variable-locale applications
1074 to be written
- 1075 • whether the specification requires the implementation to provide explicit support for
1076 variable-locale applications, for example by tagging information with locale identifiers.

1077 Except for some cases where X/Open can not change the specifications unilaterally (identified as
1078 issues of applications portability), the X/Open interworking specifications considered in this
1079 technical study do require implementations to allow variable-locale applications to be written.
1080 However, they do not require implementations to provide explicit support for variable-locale
1081 applications. This means that interworking applications must make explicit provision for
1082 variable-locale operation where this is required.

1083 The situation for multi-locale applications is similar. Except in some cases where X/Open can
1084 not change the specifications unilaterally, the X/Open Interworking Specifications considered in
1085 this technical study do not contain features that could be implemented in a way that would
1086 prevent multi-locale applications from being written, but they do not require implementations to
1087 provide explicit support for such implementations. Interworking applications must therefore
1088 make explicit provision for multi-locale operation where this is required.

1089 In general, the X/Open interworking specifications:

1090 • can be used by single locale applications that do not use codesets (such as UNICODE or ISO
1091 10646) that allow embedded nulls (and changes that would enable such codesets to be used
1092 will be considered)

1093 and

1094 • can be used by variable-locale and multi-locale applications (with the same restriction on the
1095 codesets that they can use)

1096 but

1097 • do not provide explicit support (for example, by attaching locale identifiers to information)
1098 for variable-locale and multi-locale applications.

1099 The exceptions to this (assuming that the change described in Chapter 7 of this technical study is
1100 incorporated in the XTI specification) are listed below:

1101 • XFTAM provides non-locale-dependent diagnostic strings (but this is in accordance with the
1102 FTAM standard)

1103 • XBSFT has an English-language user interface

1104 • some of the OM Attributes of the XOM, XFTAM, BSFT, X.400, XMS and XDS APIs represent
1105 character strings, but the ASN.1 string syntaxes that can be used are not sufficiently general
1106 to cater for all possible codesets (but this is in accordance with the underlying OSI standards)

1107 • the Directory to which the XDS provides an interface carries out character string
1108 comparisons that are not locale-dependent (but this is in accordance with the underlying
1109 X.500 standards)

1110 • NetBIOS and SMB implicitly assume an ASCII codeset and this has implications for the XTI,
1111 SMB and IPC for SMB specifications.

Change Requests for Internationalisation

1112

1113 This chapter contains the formal change requests for the X/Open interworking specifications
1114 that were originally raised in the previously published X/Open **Interworking**
1115 **Internationalisation** snapshot. This chapter also describes how each of these change requests
1116 has since been resolved.

1117 These change requests address the issues that could be addressed by X/Open in isolation or in
1118 co-operation with the X.400 API Association. They do not address issues requiring
1119 modifications to international standards or consultation with the PC supplier community.

1120 7.1 The XTI Specification

1121 The following changes to the XTI specification were proposed.

1122 Document: The XTI Specification
1123 X/Open CAE Specification, C196 or XO/CAE/91/600 (January 1992).

1124 Change number: XOP-1

1125 Source: X/Open / I18n CR from I18n of Interworking Specs SS

1126 Title: Internationalised Error Messages
1127 Functional Upgrade

1128 Qualifier: Major Technical

1129 Rationale: The functions *t_error()* and *t_strerror()* return strings. For international
1130 operation, these should be in the natural language of the locale established
1131 by the application program.

1132 Change:

- 1133 i. In the man page for *t_error()*, change the description as follows:
- 1134 a. Remove the text “language-dependent” from the first line.
 - 1135 b. In the first line of the third paragraph, replace the text
1136 “implementation-defined.”
1137 with
1138 “dependent on the current locale.”
- 1139 ii. In the man page for *t_strerror()*, replace the text in the description:
1140 “language-dependent error message string”
1141 with
1142 “message string based on the current locale”
- 1143 Also, replace the text:
1144 “implementation-defined”
1145 with
1146 “the natural language based on the current locale”

1147 Document: The **XTI** Specification
 1148 X/Open CAE Specification, C196 or XO/CAE/91/600 (January 1992).

1149 Change number: XOP-2

1150 Source: X/Open / I18n CR from I18n of Interworking Specs SS

1151 Title: Internationalised Error Messages
 1152 Functional Upgrade

1153 Qualifier: Major Technical

1154 Rationale: Some XTI functions return pointers and character strings as arguments to
 1155 function calls or the return value from a function call. For internationalised
 1156 operation, this should be in the natural language of the locale established by
 1157 the application program.

1158 Change:

1159 i. In the man page for `t_error()`, replace:

1160 `char *errmsg`

1161 with:

1162 `wchar_t *errmsg`

1163 ii. In the man page for `t_strerror()`, replace:

1164 `char *t_strerror`

1165 with:

1166 `void *t_strerror`

1167 Change Request (CR) XOP-1 was accepted (with a minor modification to the wording). It has
 1168 not yet been implemented, but will be implemented in the next published version of the **XTI**
 1169 specification.

1170 CR XOP-2 was withdrawn, for reasons discussed in Chapter 6, and X/Open will now consider
 1171 the addition of parallel interfaces using **wchar_t** rather than simply replacing existing interfaces
 1172 that use **char**.

1173 **7.2 The XAP, XAP-TP and XAP-ROSE Specifications**

1174 The following changes to the XAP specification were proposed:

1175 Document: The XAP Specification
1176 X/Open Preliminary Specification, P203 (June 1992).

1177 Change number: XOP-3

1178 Title: Internationalised Diagnostic Messages

1179 Qualifier: Minor Technical

1180 Rationale: Function *ap_get_env()* returns a diagnostic message in field error of structure
1181 *ap_diag_t* when passed the AP_DIAGNOSTIC attribute. For
1182 internationalised operation, this should be in the natural language of the
1183 locale established by the application program. To enable the character set of
1184 any natural language to be used, and to allow for the encoding scheme of ISO
1185 10646, a null-terminated character string should not be used.

1186 Change:

1187 i. In the ENVIRONMENT man pages definition of type *ap_diag_t*, (page
1188 54) and in the repeat definition in Appendix A (page 195), change:

1189 `char *error /* textual message */`

1190 to:

1191 `wchar_t *error /* textual message */`

1192 ii. Change the ENVIRONMENT man pages description of the error field
1193 (near the bottom of page 55) to:

1194 “The error field will be set up to point to a text string, in the natural
1195 language of the current locale, which describes the error condition, or
1196 will consist of a single null character if no such text string is available.”

1224 **7.3 The XOM Specification**1225 The following change to the **XOM** specification was proposed:1226 Document: The **XOM** Specification
1227 X/Open CAE Specification, C180 or XO/CAE/91/080 (November 1991).

1228 Change number: XOP-5

1229 Title: Support for Universal and Unrestricted Character Strings

1230 Qualifier: Major Technical

1231 Rationale: The character string types supported by XOM are not sufficiently general to
1232 support all character sets and codesets used internationally. This issue has
1233 been recognised by the standards bodies responsible for ASN.1. A new DIS
1234 8824 is currently in ballot. It includes (among other changes from the present
1235 version) support for two new string types. These are:1236 Universal String which supports the Universal Multi-Octet Coded
1237 Character Set of ISO DIS 106461238 Unrestricted String which supports any character set and encoding
1239 scheme, provided they have OIDs.1240 These string types are sufficiently general for full international use. It would
1241 be possible to make the changes to XOM that would add support for these
1242 types independently of any other changes that might be suggested to reflect
1243 other changes that have been made to ASN.1

1244 Change:

1245 i. In Section 3.4, Table 2, add, after "UTC Time":

1246 "Universal²"1247 "Unrestricted⁴"

1248 and add the following footnote:

1249 "⁴ Values of this syntax are represented in their BER encoded form."

1250 ii. In Section 3.8, Table 5, add, after the "Teletex String" line:

1251 "Universal String String(Universal)"

1252 "Unrestricted String String(Unrestricted)"

1253 iii. In Section 4.2.12, replace:

1254 "A zero character follows"

1255 with:

1256 "Universal and Unrestricted strings can contain null octets. Only the
1257 length-specified form shall be used to represent strings of these types.
1258 When either form is used, a null character (that is, an instance of the
1259 character whose encoding is zero) follows"

1260 In Section 4.2.13, add to clause 4, after:

1261 "teletex string"

1262 the following:

1263 “universal string, unrestricted string”.

1264 iv. In Section 4.5, add to the `/* Syntax */` section, after the definition of

1265 OM_S_TELETEX_STRING, the following:

1266 `#define OM_S_UNIVERSAL_STRING ((OM_syntax)28)`

1267 `#define OM_S_UNRESTRICTED_STRING ((OM_syntax)29)`

1268 Change Request (CR) XOP-5 was accepted and has been implemented (with minor changes) in

1269 the referenced **XOM X/Open CAE** specification.

1270 **7.4 The BSFT Specification**

1271 The following change to the **BSFT** specification was proposed:

1272 Document: The **BSFT** Specification
1273 X/Open CAE Specification, C194 (December 1991).

1274 Change number: XOP-6

1275 Title: International Support for User Interfaces

1276 Qualifier: Major Technical

1277 Rationale: The user interface for BSFT is defined to use only the English language. This
1278 restricts the use of BSFT internationally.

1279 Change: In Appendix A, immediately after the first paragraph of the Appendix ("This
1280 section defines the user interface for the BSFT facility."), add the following
1281 paragraphs:

1282 "This section specifically defines an English language user interface. An
1283 implementation may, however, support user interfaces in other languages."

1284 "In each user interface supported, other than the one specifically defined in
1285 this section, there shall be commands, parameters and responses that are
1286 equivalent to all those that are defined in this section. How the commands,
1287 parameters and responses of each user interface correspond to those defined
1288 in this section shall be documented."

1289 "If an implementation supports more than one user interface, and one of the
1290 environment variables LC_ALL, LC_MESSAGES, or LANG is set when the
1291 BSFT facility is invoked, and a user interface appropriate to the locale
1292 referenced by that environment variable is supported, then that user
1293 interface shall apply. If more than one of those environment variables are
1294 set, then the order of precedence shall be LC_ALL first, LC_MESSAGES
1295 second, and LANG third."

1296 "It should be noted that a user who attempts to transfer a file that has been
1297 defined using language and cultural conventions other than those of his
1298 current locale may experience problems, and that the BSFT facility does not
1299 provide a means for the user to determine the locale in which a file (on either
1300 the local or the remote system) has been defined, or to determine or influence
1301 the locale assumed by the BSFT responder."

1302 This CR was rejected, with the rationale that internationalisation of the BSFT user interface
1303 should be addressed as part of the solution of the larger issue of internationalising the X/Open
1304 commands.

1305 **7.5 The XFTAM Specification**1306 The following change to the **XFTAM** specification was proposed:1307 Document: The **XFTAM** Specification
1308 X/Open Preliminary Specification, P206 (September 1992).

1309 Change number: XOP-7

1310 Title: Internationalised Error Messages

1311 Qualifier: Minor Technical

1312 Rationale: Function *gpperror()* returns two strings. For internationalised operation, these
1313 should be in the natural language of the locale established by the application
1314 program. To enable the character set of any natural language to be used, the
1315 requirement that they must be printable strings should be dropped. (The
1316 proposed change would allow any character string type representable in
1317 XOM. In conjunction with CR XOP-5, proposed for XOM, this would allow
1318 Universal and Unrestricted Strings).1319 Change: In the man page for *ft_gpperror()*,1320 i. In the description of *Return_string*, change:1321 “Return_string(OM_String(Printable))
1322 The printable string ... NULL-terminated.”

1323 to:

1324 “Return_string(OM_String(*))
1325 A text string in the natural language of the current locale that
1326 represents the Return-Code attribute of the *API_out_in* parameter.
1327 This is the XFTAM-specified error code and is always returned. The
1328 resulting string is formatted for printing as a self-contained unit (for
1329 example, in an English language locale, it includes a terminating
1330 newline character).”1331 ii. In the description of *Vendor_string*, change:1332 “Vendor_string(OM_String(Printable))
1333 The printable string ... NULL-terminated.”

1334 to:

1335 “Vendor_string(OM_String(*))
1336 A text string in the natural language of the current locale that
1337 represents the Vendor-Code attribute of the *API_out_in* parameter.
1338 This is an optional implementation-specific error code and shall not
1339 be returned if the *API_out_in* parameter did not contain an
1340 equivalent code. The resulting string is formatted for printing as a
1341 self-contained unit (for example, in an English language locale, it
1342 includes a terminating newline character).”1343 CR XOP-7 was accepted and has been implemented in the referenced **XFTAM** X/Open CAE
1344 Specification.

1345 **7.6 The X.400 API Specification**1346 The following change to the **X.400 API** was proposed:1347 Document: The **X.400 API** Specification
1348 X/Open CAE Specification, C191 or XO/CAE/91/100 (December 1991).

1349 Change number: XOP-8

1350 Title: Printable Strings Used Internationally

1351 Qualifier: Minor Technical

1352 Rationale: Following the 1988 version of the X.400 Series Recommendations, the **X.400**
1353 **API** requires use of Printable Strings for certain O/R Address attributes
1354 when the API is used to send messages internationally. The corresponding
1355 requirement on the protocol was dropped from ISO 10021 and is expected to
1356 be dropped from the 1992 version of the X.400 Series Recommendations.

1357 Change: In Section 5.2.31, delete the first and last sentences of note 3, that is:

1358 a. delete "If only one value ... String(Printable)."

1359 b. delete "Printable strings are required internationally ...
1360 communication."1361 CR XOP-8 was accepted and has been implemented in the referenced **X.400 API, Issue 2** X/Open
1362 CAE Specification.

1363 **7.7 The XDS Specification**

1364 The following change to the XDS API was proposed:

1365 Document: The XDS Specification
1366 X/Open CAE Specification, C190 or XO/CAE/91/090 (November 1991).

1367 Change number: XOP-9

1368 Title: Support for Directory Strings

1369 Qualifier: Major Technical

1370 Rationale: Following X.520 (1988), XDS defines a number of Directory Attributes to
 1371 have syntax String(Teletex). This is not sufficiently general to support all
 1372 character sets and codesets used internationally. This issue has been
 1373 recognised by the standards bodies responsible for X.520. A new version of
 1374 X.520 has been proposed and (in this respect at least) has been agreed from a
 1375 technical point of view, but has not yet been formally adopted by the CCITT.
 1376 This introduces a new syntax, "Directory String", for these attributes.
 1377 Assuming that CR XOP-5 is accepted for XOM, it would be possible to make
 1378 the changes to XDS that would add support for this syntax, and to make
 1379 them independently of any other changes that might be suggested to reflect
 1380 other changes that are made to the X.500 Series recommendations.

1381 Change:

1382 i. In Section 1.1, add, after "1988", a superscript 1 and place the following
 1383 footnote at the bottom of the page:1384 " ¹ It also takes account of some changes made in the 1992 version of
 1385 these standards."

1386 ii. In Section 7.2, add, after:

1387 "C constants start with DS_A.)"

1388 a new paragraph:

1389 "Several of the attribute types are defined in the 1992 version of the
 1390 Standards to have ASN.1 syntax DirectoryString. This is a CHOICE of
 1391 TeletexString, PrintableString and UniversalString. In these cases, the
 1392 values of the corresponding Attribute-Values OM attributes can have
 1393 syntaxes String(Teletex), String(Printable) or String(Universal). This is
 1394 indicated by describing their syntaxes as String(Directory)."

1395 iii. In Section 7.2, 4th paragraph:

1396 a. Change

1397 "two general rules" to "three general rules"

1398 b. Add, at the end of the paragraph:

1399 "For all attribute values whose syntax is indicated as
 1400 String(Directory), differences in the case of alphabetical
 1401 characters shall be considered insignificant and, if the strings
 1402 being compared are of different syntax, the comparison shall
 1403 proceed as normal so long as the corresponding characters are in
 1404 both character sets, but shall fail otherwise."

- 1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
- iv. In Section 7.2, Table 34, in the entries for A-Business-Category, A-Common-name, A-Description, A-Knowledge-Information, A-Locality-Name, A-Organisation-Name, A-Organisational-Unit-Name, A-Physical-Delivery-Office-Name, A-Post-Office-Box, A-Postal-Code, A-State-Or-Province-Name, A-Street-Address, A-Surname, A-Title, change:
 “String(Teletex)”
 to:
 “String(Directory)”
 (This should remove all occurrences of String(Teletex)" from the table.)
 - v. In Section 7.12, change the Value Syntax of Postal-Address from:
 “String(Teletex)”
 to:
 “String(Directory)”.
Note: Since Teletex String is a special case of Directory String, the uses of DS_A_COMMON_NAME etc. In the programming examples of XDS Chapter 9 need not be changed.
- CR XOP-9 was accepted and has been implemented in the referenced XDS, Issue 2 X/Open CAE Specification.

1424 **7.8 The XNFS Specification**

1425 No change requests were proposed for this specification.

1426 **7.9 The (PC)NFS Specification**

1427 No change requests were proposed for this specification.

1428 **7.10 The SMB Protocols Specification**

1429 No change requests were proposed for this specification.

1430 **7.11 The IPC Mechanisms for SMB Specification**

1431 No change requests were proposed for this specification.

1432 / *X/Open Technical Study*

1433 **Part 3**

1434 **X/Open Data Management Specifications**

1435 *X/Open Company Ltd.*

Introduction

1436

1437 The chapters in this part of this technical study consider the impact of internationalisation on the
1438 X/Open data management specifications. These specifications are at present either CAE
1439 specifications or are expected shortly to become preliminary specifications. They consist of the
1440 following documents (full details are given in **Referenced Documents** on page xv).

- 1441 • the **SQL** specification
- 1442 • the **CLI** specification
- 1443 • the **RDA** specification.

1444 **Structure of This Part**

1445 Chapter 9 discusses general internationalisation issues that are associated with SQL, as defined
1446 by the ISO standards. Many of these issues are also common to the X/Open data management
1447 specifications.

1448 Chapter 10 examines the implications of internationalisation on the data management
1449 specifications listed above.

1450 After this, Chapter 11 presents conclusions and recommendations.

Structured Query Language (SQL)

1451

1452 9.1 Overview

1453 The X/Open data management specifications (**SQL**, **CLI** and **RDA**) are all related to Structured
1454 Query Language (SQL). This chapter discusses the internationalisation issues associated with
1455 SQL in general. It is a prelude to Chapter 10 which discusses the X/Open data management
1456 specifications individually.

1457 SQL has been standardised by ISO. The most recently approved SQL standard is ISO/IEC
1458 9075: 1992; this is sometimes called SQL 92 or SQL2 (see the referenced ISO SQL 92 standard).
1459 This technical study uses the term SQL 92 when referring to this standard. The standard
1460 specifies the syntax and semantics of SQL, for use where SQL statements are used in the
1461 following ways:

- 1462 • invoked directly (for example from a user interface)
- 1463 • stored as procedures that can be called from programs
- 1464 • embedded in programs written in other programming languages.

1465 For direct invocation, SQL 92 does not define how the statements are invoked or how any results
1466 are returned. For calling SQL procedures from, or embedding SQL statements in, a
1467 programming language, SQL 92 includes syntax for the particular programming languages Ada,
1468 C, COBOL, Fortran, MUMPS, Pascal and PL/I.

1469 SQL 92 defines three levels of conformance:

- 1470 • Entry Level SQL
- 1471 • Intermediate SQL
- 1472 • Full SQL.

1473 Intermediate SQL is a subset of Full SQL, and Entry Level SQL is a subset of Intermediate SQL.

1474 Work is currently proceeding on a new version of the SQL standard, currently referred to as
1475 SQL3. This is documented in the referenced ISO SQL3 draft standard, which incorporates the
1476 following significant new features:

- 1477 • active “rules”, called triggers
- 1478 • abstract data types
- 1479 • multiple null states
- 1480 • PENDANT referential integrity
- 1481 • a recursive union operation for query expressions
- 1482 • enumerated and boolean data types
- 1483 • SENSITIVE cursors.

1484 The SQL3 draft standard does not include any significant features related to internationalisation
1485 beyond those already contained in Full SQL 92.

1486 9.2 Multiple Character Sets

1487 Character strings constitute one of the types of data that can be stored in relational databases
 1488 and manipulated using SQL. SQL provides means for entering and retrieving character string
 1489 data, and for sorting and ordering character string data.

1490 Clearly, users may wish to use character string data that relates to any language and cultural
 1491 environment. They may even wish to store, in the same table, character string data that relates
 1492 to several different language and cultural environments. For example, a user might wish to use
 1493 a table to store a multi-lingual glossary, with columns containing equivalent terms and
 1494 definitions in English, French, Russian, Arabic and Japanese. The same user might then wish to
 1495 derive an appropriately sorted copy of the table for each of these languages.

1496 If users are to be able to carry out such operations, SQL must allow:

- 1497 • the use of the appropriate character sets
- 1498 • several different character sets to be used in a single table
- 1499 • the sorting operation to use the collation order appropriate to the character set or sets of the
 1500 data to be sorted.

1501 Generally, these conditions are satisfied by Full SQL (as defined in SQL 92), which is rich in
 1502 features relating to international use. The conditions are however only partly satisfied by
 1503 Intermediate SQL, and they are not satisfied at all by Entry Level SQL. In particular:

- 1504 • Full SQL and Intermediate SQL, but not Entry Level SQL, allow *character set specifications* that
 1505 allow an application to use several different sets of characters.
- 1506 • Full SQL, but not Intermediate SQL or Entry Level SQL, also allows *collation definitions*,
 1507 which allow an application to specify collating sequences other than the default collating
 1508 sequence for a character set.

1509 9.3 Use of Standard Names

1510 Although Full SQL has features that allow an application to specify multiple character sets and
 1511 collating sequences, it does not require that the implementation supports standard identifiers for
 1512 them. So, for example, one implementation might support a character set called **ISO8859-1**,
 1513 while another might support the same character set but call it **ISO1**. This clearly inhibits
 1514 portability of applications.

1515 This problem would be overcome if there were a set of standard character set identifiers which
 1516 all SQL implementations must use. It would be advantageous if these identifiers were the ones
 1517 defined for locales in the registry proposed in the **DISS Issue 1** snapshot. An application could
 1518 then make SQL statements such as:

```
1519 CREATE TABLE foo (
1520     col_a CHARACTER (10) CHARACTER SET ge_GE.8859-1
1521                               COLLATE ge_GE.8859-1@foobar,
1522     col_b CHARACTER (20) COLLATE fr_FR.8859-1
1523 )
```

1524 and be assured of portability across a wide range of implementations.

1525 Failing the definition of standard names for the whole user community, functional profiles and
 1526 corporate or national procurement standards may define their own sets of names to ensure
 1527 portability. For example, the FIPS 127-2 standard defines the character set names LATIN1,
 1528 ASCII_FULL, and ASCII_GRAPHIC.

1529 9.4 Character Set Not Determined by Locale

1530 In order to write a fully internationalised application — that is, an application that exploits all
1531 the features of SQL and which can be used without re-compilation in any language and cultural
1532 environment — it is necessary that the character set assumed by the SQL processor in direct
1533 execution and dynamic execution of SQL statements can be determined by the locale
1534 mechanism.

1535 SQL 92 does not refer to locale mechanisms. Instead it provides the SET NAMES statement
1536 which allows the application programmer to nominate the default character set for dynamic
1537 SQL statements and for direct invocation of SQL statements. This statement is only available in
1538 Full SQL.

1539 An application could thus:

- 1540 • determine the codeset of the current locale by calling *nl_langinfo*(CODESET)
- 1541 • use the SQL SET NAMES statement to set the default character set names for identifiers and
1542 character string literals in preparable statements.

1543 The problem is that the application must be able to convert the codeset name obtained by means
1544 of *nl_langinfo*() into the character set name required by SQL. This cannot be done in a portable
1545 way (except in the context of a local standard for such names, such as that defined in FIPS 127-2),
1546 since neither **XPG4** nor the SQL 92 standard defines standard character set names.

1547 An internationalised program that uses locales may make use of the features of Full SQL to run
1548 in different language and cultural environments, but is not able to achieve full
1549 internationalisation in a portable way. The character set and collation features are however not
1550 available in Entry Level SQL, and only some of them are available in Intermediate SQL. It is
1551 hard to see how an application can have any significant degree of internationalisation if only the
1552 facilities of Entry Level SQL are available.

1553 Multiple character set facilities (as in Full SQL) and locales (as in **XPG4**) are not the only ways in
1554 which internationalisation could be provided. For example, each item of data could be *self-*
1555 *announcing*, that is, could contain an indication of the language and cultural environment in
1556 which it should be processed. A mechanism to support this is described in the **DISS Issue 1**
1557 snapshot. Such a mechanism could be used in conjunction with SQL — even with Entry Level
1558 SQL. The use of such a mechanism in conjunction with SQL forms no part of current SQL
1559 standardisation work, however.

1560 9.5 Encodings

1561 SQL 92 does not specify how character (or other) data is represented internally by an SQL
1562 implementation, nor does it say how data is represented when passed between the SQL
1563 implementation and other system components. However, SQL 92 does attach semantics to the
1564 data. For example, each item of character data is understood to consist of a set of characters
1565 from some known character set. Clearly, other components of a system that includes SQL must
1566 attach the same semantics to the data as the SQL component. This means that the components
1567 must all use the same character set encodings.

1568 **Direct Invocation**

1569 In direct invocation, the manner in which the statements are invoked and the results are
1570 returned (and hence the manner in which data is passed) is implementation-defined. This means
1571 that vendors must document the character sets and encodings (the *codesets*) that they can accept,
1572 so that the SQL implementation can be used in conjunction with other products.

1573 **Non-direct Invocation**

1574 In other cases, the system components that pass data to or receive data from an SQL
1575 implementation are typically:

- 1576 • the text editors used to create SQL statements
- 1577 • the host language processors (including compilers, interpreters, and run-time libraries) that
1578 process host language programs that call SQL procedures, or in which SQL statements are
1579 embedded.

1580 **Integration with Text Editors**

1581 The issue of text editors (the term is used here to include text processing programs of all types)
1582 arises when an SQL application refers to particular character strings. Such strings may be used
1583 as names (table names, column names and so on), or as application data (for example, in
1584 WHERE clauses of SELECT statements). If such strings are to be meaningful, the SQL
1585 implementation must use the same encoding scheme as the text editor. Literal character strings
1586 are likely to be specific to a particular language, however, and their use for applications data
1587 should therefore be avoided in internationalised applications. This might be done, for example,
1588 by using character string variables whose values are obtained from message catalogues.

1589 **Integration with Host Language Processors**

1590 Data can be passed between SQL and a host language processor in several ways: for example, in
1591 the parameters of SQL procedures, or in variables of embedded SQL statements.

1592 A programmer can hard code character encodings into a program, for example, with the C
1593 statement:

```
1594        c = (char) 65;
```

1595 The use of such constructs should however be avoided when writing internationalised software,
1596 as discussed in the referenced **Internationalisation Guide**. If the advice given in the
1597 **Internationalisation Guide** is followed, the character encodings are determined entirely by the
1598 host language processor and the currently established locale.

1599 Although SQL 92 requires an implementation to state which character sets it can handle, SQL 92
1600 does not clearly state a requirement for implementations to describe the encodings that are used
1601 to represent the characters of those character sets. For example, an implementation could
1602 support an English character set, without saying whether it is encoded using ASCII, EBCDIC,
1603 UNICODE, or some other encoding scheme. This information is needed to determine whether a
1604 particular implementation of SQL and a particular host language processor can interwork. It
1605 is therefore important that an SQL implementation should specify the character set encodings that
1606 may be used by a host language processor for procedure parameters, embedded variables and
1607 executable SQL statements.

1608 9.6 String Operations

1609 The use of encodings such as UNICODE and ISO 10646 in which a single graphic symbol may be
 1610 represented by a combination of several members of the codeset raises questions with regard to
 1611 string comparisons and related operations. These concern whether a graphic symbol represented
 1612 by a combination of several codeset elements should be treated as a single character or as
 1613 multiple characters for collation purposes. SQL 92 appears to treat such a combination as
 1614 several characters rather than one⁵.

1615 Although the use of combining characters in UNICODE is a particularly topical instance of this
 1616 issue, it also arises for other character set encodings. Spanish, for example, requires the
 1617 character **ch** to collate as a single entity. This character does not have a single codeset element in
 1618 ISO Latin-1, and the collation mechanism must take account of this.

1619 As far as collation operations are concerned, collating sequences are regarded in SQL 92 as
 1620 operating on strings rather than on individual characters. However, the default collating
 1621 sequences for standard character repertoires and standard universal forms of use are based on
 1622 the numerical order of codeset elements; they process strings character by character. Also, such
 1623 orders have no intended relationship to natural language collation orders, so they may not
 1624 produce the desired ordering. This may lead to counter-intuitive results in some cases. For
 1625 example, a string containing lower case <e> followed by an acute accent combining character
 1626 would collate differently from one that contains a single <e with acute accent> character, and
 1627 neither collation would be that required by French, where <e with acute accent> is generally
 1628 treated for collation purposes as though it were the simple letter <e>.

1629 Counter-intuitive results may also be obtained from operations involving wildcard characters
 1630 and from those in which substrings are indicated by their numeric positions within strings. For
 1631 example, the second letter in the French string "défense de fumer" is <e with acute>. Is <f> the
 1632 third or the fourth character of that string?

1633 9.7 ISO 10646 and C

1634 The introduction of UNICODE or ISO 10646 encodings raises questions concerning the use of
 1635 SQL in conjunction with a C host program. An SQL implementation based on UNICODE or
 1636 ISO-10646 might use 2-byte or 4-byte character encodings, but these cannot conveniently be
 1637 handled as character strings by a C host program. Moreover, SQL character data is represented
 1638 in C by null-terminated strings, while UNICODE and ISO 10646 allow character encodings that
 1639 include null bytes, as discussed in Section 2.3 on page 9. How these difficulties should be
 1640 handled is not yet clear, but possibilities include:

- 1641 • The C program encodes the characters using an encoding that is legal for C (such as that in
 1642 the ISO 8859 standard or a UTF) and the SQL implementation converts this to
 1643 UNICODE/ISO 10646.

1644 _____
 1645 5. The recommendation by many members of the Unicode Consortium is to decompose the pre-combined characters (for example,
 1646 A-Umlaut = A + Umlaut), then conduct sorting based on letter, case, cultural and accent weights.

1647 • The C program stores characters as quantities of type `wchar_t`, but the SQL implementation
1648 treats them as arrays of type `char`. A future version of the SQL standard will require strings
1649 to be terminated by the number of null bytes appropriate to the encoding scheme (two null
1650 bytes or four null bytes for ISO 10646, as opposed to a single null byte for ASCII). A recent
1651 Erratum to the SQL 92 standard does in fact clarify that the two or four null bytes are
1652 required.

1653 These difficulties do not arise with languages such as COBOL, which do not constrain the way
1654 that character strings are encoded.

1655 **9.8 Reserved Words and Special Characters**

1656 SQL 92 defines a number of reserved words (ABSOLUTE, ACTION, ADD and so on) and gives a
1657 special importance to characters such as double quote, percent and ampersand. These have an
1658 English language flavour in that the reserved words are meaningful in English. The usage of
1659 some of the special characters is similar to their usage in English (for example, the use of quote
1660 characters to begin and end a character string literal; many languages use different characters for
1661 these purposes). This is undoubtedly an inconvenience to anyone developing an SQL
1662 application in a non-English environment. However, it does not affect the degree to which an
1663 SQL application can be internationalised. It is probably acceptable, given the technical
1664 difficulties of implementing a multi-lingual or internationalised version of SQL.

1665 **9.9 Numeric and Date Literals**

1666 SQL 92 defines character string representations of numbers and dates (*numeric literals* and *date*
1667 *literals*) that use the period character as a decimal separator, and have a year-month-day format
1668 for dates. These are not the natural representations in all cultural environments. In an
1669 internationalised application, they would have to be converted to a locale-dependent format
1670 before being displayed at the user interface.

1671 **9.10 Diagnostic Information**

1672 The GET DIAGNOSTICS statement specified by SQL 92 can include character strings
1673 representing alpha-numeric codes (SQLSTATE) and implementation-defined character strings
1674 (MESSAGE_TEXT). The SQLSTATE strings, which are limited to using the 10 digits and the 26
1675 upper-case Latin alphabetic characters, are a possible inconvenience to application developers in
1676 some language and cultural environments, but do not affect the degree to which applications
1677 can be internationalised. Applications that use implementation-defined MESSAGE_TEXT
1678 strings may be dependent on the language in which those strings are written. Internationalised
1679 applications should not use such strings as a source of error message text.

1680 SQL 92 says that when MESSAGE_TEXT is requested by the application, an implementation
1681 may set the string returned to spaces, to a zero length string, or to a character string describing
1682 the condition indicated by the returned SQL_STATE value. If implementations follow this
1683 precept (which is in a non-normative note) then applications should not need to use
1684 MESSAGE_TEXT strings since the relevant information should also be given by SQL_STATE. Of
1685 course, it is possible that an implementation could return a string obtained from a message
1686 catalogue and determined by the current locale. There is no requirement on implementations to
1687 do this; an application that relied on such behaviour would not be portable.

1688 **9.11 Arithmetical Expressions**

1689 The English representation of arithmetical expressions and numbers may not be appropriate for
1690 all language and cultural environments. For example, some environments may use a notation
1691 other than the “arabic” one for representing numbers. This implies a requirement for the
1692 application to convert such information to the form required by the local language and cultural
1693 environment, but does not prevent an application that performs such a conversion from being
1694 internationalised. It should be noted, however, that the locale facilities currently specified by
1695 X/Open do not provide for differing arithmetical conventions.

1696 **9.12 Directionality**

1697 SQL 92 (and other SQL specifications) appear to assume a left to right and top to bottom, row-
1698 wise, display scheme. Such a scheme is not appropriate to languages (like Hebrew and Arabic)
1699 which scan right to left, or to languages that scan not only right to left but also column-wise
1700 rather than row-wise.

1701 In fact, these issues are concerned with the way that information is presented at the user
1702 interface, rather than how it is organised in the database. A *row* can be presented in any
1703 direction: left to right, right to left, top to bottom or bottom to top. SQL 92 does not specify how
1704 information is to be presented at the user interface; furthermore none of the X/Open data
1705 management specifications are concerned with the user interface. These issues are therefore not
1706 relevant to this technical study.

1707

1708 10.1 The SQL Specification**1709 10.1.1 Overview**

1710 The X/Open **SQL** specification defines the application programming interface for X/Open-
1711 compliant relational database management systems. It includes almost all of the Entry Level
1712 provisions of SQL 92 relating to embedding SQL in C and COBOL host language programs. It
1713 also includes some features (in particular, Dynamic SQL) from the Intermediate and Full levels
1714 of SQL 92.

1715 10.1.2 Internationalisation Implications

1716 The Intermediate and Full level SQL features in the X/Open **SQL** specification do not include
1717 the multiple character set and collation sequence features. All of the issues identified for Entry
1718 Level SQL in Chapter 9 of this technical study therefore apply to the **SQL** specification. There
1719 are no additional issues.

1720 10.2 The CLI Specification

1721 10.2.1 Overview

1722 The CLI specification describes an API for database access that is an alternative to the API
1723 defined in the X/Open SQL specification. The SQL specification describes how to create SQL
1724 statements that can be embedded in a C or COBOL source program and, after suitable pre-
1725 processing, can be compiled by a C or COBOL compiler. This approach is not always the best
1726 one. The CLI specification defines C functions and COBOL subroutines that can be called from
1727 C or COBOL programs and that will provide the functionality that is provided by the SQL
1728 Specification. It also provides functions and subroutines that enable the programmer to control
1729 connections between database clients and servers.

1730 Note that this is a different approach from the use of procedures as defined in SQL 92.
1731 Procedures are SQL statements that can be called from a host program, whereas the CLI
1732 comprises C or COBOL routines that cause SQL statements to be executed.

1733 10.2.2 Internationalisation Implications

1734 General Implications

1735 As for the SQL specification, all the issues identified for Entry Level SQL in Chapter 9 of this
1736 technical study apply to the CLI specification.

1737 Passing of Character Strings

1738 There is a difference between the CLI specification and the SQL specification in the way that
1739 character strings are passed between the host language processor and the SQL implementation.

1740 For the SQL specification:

- 1741 • character strings are passed in variables
- 1742 • dynamic SQL statements are passed in variables
- 1743 • other SQL statements are embedded in the host source program (and so are part of the input
1744 to the SQL pre-processor).

1745 For the CLI specification:

- 1746 • all SQL statements and character string data are passed as function and subroutine
1747 arguments — for example, the *vcSqlStr* argument of *prepare()* and the *vcColName* argument of
1748 *DescribeCol()*.

1749 The same considerations apply to such arguments as apply to character string variables in
1750 embedded SQL. In particular, there are the same problems with UNICODE/ISO 10646 and null
1751 terminated strings in C. Thus, this difference between the CLI specification and the SQL
1752 specification introduces no new internationalisation issues.

1753 **Character String Conversions**

1754 CLI provides for automatic conversion between numeric values and character strings. The
1755 application can supply or retrieve a numeric value in the form of a character string; the
1756 implementation performs the conversion to or from numeric format. The character string
1757 representation of numeric values is that defined for numeric literals in SQL 92. It is not the
1758 natural representation in all language and cultural environments but, as discussed in Chapter 9
1759 of this technical study, its use does not prevent the internationalisation of applications.

1760 **10.3 The RDA Specification**1761 **10.3.1 Overview**

1762 The **RDA** specification defines the format for remote communications with an SQL database. It
 1763 is based on the ISO RDA standard, and describes OSI Application Protocol Data Units (APDUs)
 1764 and their use in conjunction with the Association Control Service Element (ACSE) and the
 1765 Presentation and Session services.

1766 **10.3.2 Internationalisation Implications**1767 **General Implications**

1768 The **RDA** specification does not impose constraints on the types of SQL statement that can be
 1769 executed remotely. The issues pertaining specifically to Intermediate and Entry Level SQL
 1770 therefore do not apply.

1771 The **RDA** specification states the following:

- 1772 • SQL statement text and SQL character string data are represented as octet strings
- 1773 • object identifiers for their encodings are associated with them.

1774 The issues pertaining to locale-dependence of the character sets and to the definition of the
 1775 encodings of such character data therefore do not apply to it (but see **Visible Strings** (below) for
 1776 issues pertaining to the encodings of other character data).

1777 The other issues pertaining to Full SQL apply.

1778 **Visible Strings**

1779 The **RDA** specification requires that certain information is represented by ASN.1⁶ *Visible Strings*.
 1780 This effectively means that the information must be encoded in basic ASCII. The information
 1781 concerned includes *diagnosticInformation* (various services), *identityOfUser* (R-Initialise), *aborted*
 1782 (R-Status) *dataResourceName* (R-Open), and *colName*, *classOrigin*, *subclassOrigin*, *messageText*,
 1783 *sQLState*, *sQLErrorText* (R-ExecuteDBL).

1784 In the case of *sQLState*, which contains formally coded information, this is an inconvenience to
 1785 the application developer, but nothing worse (see the discussion of SQLSTATE in Section 9.10 on
 1786 page 68 of this technical study).

1787 In other cases, the developer of an internationalised application, or of an application in a
 1788 language and cultural environment other than English, is hampered to a more serious extent.
 1789 The developer may, for example, wish to display column names at the user interface of the client
 1790 system. The names typically include non-ASCII characters. Therefore they cannot be passed to
 1791 the server, because they cannot be encoded as Visible Strings. It would, of course, be possible to
 1792 define a second set of column names that use only ASCII characters, and to translate them to the
 1793 non-ASCII names before displaying them at the user interface, but this is an overhead which is
 1794 undesirable.

1795 _____

1796 6. Abstract Syntax Notation 1 (ASN.1) is a formal notation for describing information types. It is used to describe the types of
 1797 information conveyed by the OSI presentation service. It is defined in the referenced ASN.1 standard.

1798 With information such as *sQLErrorText*, which is generated by the server and returned to the
1799 client, the situation is more serious still as the application may not be able to determine the
1800 corresponding text for the user's language and cultural environment; it is therefore unable to
1801 display the information.

1802 These issues apply not only to the **RDA** specification but also to the ISO RDA standard on which
1803 it is based.

1804 Work is proceeding in ISO (ISO/IEC JTC1/SC21) on the addition to ASN.1 of base types dealing
1805 with ISO 10646. Once this work is stable, it would be possible for the **RDA** specification and the
1806 ISO RDA standard to refer to it. However, the following points must be considered:

- 1807 • it would mean that data that already exists and is encoded using some other character set,
1808 such as that in the ISO 8859 standard, would have to be converted to the ISO 10646 standard
1809 form
- 1810 • it would not enable collating sequences, conversions, and other information pertinent to the
1811 language and cultural environment to be identified.

Conclusions and Recommendations

1812

1813 This chapter presents conclusions and recommendations regarding the internationalisation of
1814 X/Open data management specifications.

1815 11.1 Conclusions

1816 The following conclusions can be drawn from the analysis of internationalisation issues in this
1817 technical study:

1818 (C-01) The internationalisation problems associated with the X/Open data management
1819 specifications are not introduced by additions to or divergences from the related
1820 International Standards; those standards have the same problems.

1821 (C-02) A portable application that uses an implementation of the **SQL** specification or the **CLI**
1822 specification cannot in general be fully internationalised, because the character sets are
1823 not determined by the current locale. The SQL implementation need not be aware of
1824 the currently established locale. Therefore it may not correctly interpret the character
1825 encodings presented to it. This issue is discussed in Section 9.4 on page 65.

1826 (C-03) Internationalisation of applications that use implementations of the **RDA** specification
1827 is limited, because of the requirement that certain character data be represented by
1828 ASN.1 Visible Strings, which means that it is essentially restricted to being
1829 representable using ASCII. This issue is discussed under **Visible Strings** on page 74.

1830 (C-04) The use of diagnostic message text (`MESSAGE_TEXT`) to convey information to
1831 applications is inappropriate for internationalised applications. This issue is discussed
1832 in Section 9.10 on page 68.

1833 (C-05) The Entry Level SQL facilities upon which the **SQL** specification and the **CLI**
1834 specification are based are inadequate for the development of applications that handle
1835 data that have multiple languages or cultural environments. This issue is discussed in
1836 Section 9.4 on page 65.

1837 (C-06) The requirements for implementations of SQL and CLI to document the character sets
1838 and encodings (the codesets) that they can accept should be stated clearly. This issue is
1839 discussed in Section 9.5 on page 65 and Section 10.2.2 on page 72.

1840 (C-07) There are issues that require clarification relating to the use of codesets that permit a
1841 single graphic symbol to be represented by a combination of codeset elements. This
1842 issue is discussed in Section 9.6 on page 67.

1843 (C-08) There are issues relating to the use of UNICODE and ISO 10646 in conjunction with the
1844 C programming language that require clarification. This issue is discussed in Section
1845 9.7 on page 67.

1846 **11.2 Recommendations**

1847 The above conclusions lead to the following recommendations.

- 1848 • X/Open should draw the attention of standards bodies to the internationalisation problems
1849 that result from the fact that the character set in which dynamically executed SQL statements
1850 are written is not locale-dependent (see conclusion (C-02)), and should consider requiring
1851 one or both of the following solutions in X/Open-compliant systems.
- 1852 — X/Open could state in its **SQL** specification and **CLI** specification that this character set
1853 should be locale-dependent.
- 1854 — X/Open could add the SET NAMES statement to its **SQL** specification and **CLI**
1855 specification; it could also require the character set names recognised by this statement to
1856 include those of standard locales.
- 1857 • In any case, it is clearly desirable that SQL applications should be able to refer to the
1858 character sets and collation sequences of standard locales, as discussed in Section 9.3 on page
1859 64. Implementors should be encouraged to support the character sets and collation
1860 sequences of locales specified by X/Open.
- 1861 • X/Open should draw the attention of the bodies responsible for the standardisation of RDA
1862 to the internationalisation problems that result from the requirements in RDA for certain
1863 data to be represented by ASN.1 Visible Strings. It should work with those bodies to define
1864 other means of representing such data (see conclusion (C-03)).
- 1865 • The X/Open Data Management Working Group and the Joint Internationalisation Group
1866 should work with each other and with the bodies responsible for the standardisation of SQL
1867 to develop alternative methods of returning the information that implementations currently
1868 supply in the form of diagnostic message text (see conclusion (C-04)). This could include the
1869 use of locale-dependent natural language text.
- 1870 • X/Open should consider enhancing the **SQL** specification and **CLI** specification to
1871 incorporate those features of Intermediate and Full SQL that allow an application to handle
1872 data that has multiple language or cultural environments (see conclusion (C-05)). While a
1873 requirement to implement Full SQL might be considered to impose too great a burden on
1874 implementors at the present time, it might be possible to identify a subset of Full SQL that
1875 would provide sufficient internationalisation capabilities, which need not be too expensive to
1876 implement.
- 1877 • The **SQL** specification should require that implementations state clearly in their
1878 documentation which codesets are allowed to be used for:
- 1879 — embedded SQL statements
- 1880 — dynamically created executable SQL statements
- 1881 — embedded variables.
- 1882 The **CLI** specification should require that implementations state clearly in their
1883 documentation which codesets are allowed to be used for C function and COBOL subroutine
1884 arguments (see conclusion (C-06)).

- 1885
- 1886
- 1887
- X/Open should work with bodies responsible for the standardisation of codesets and of SQL to clarify the following questions (with particular attention to how UNICODE and ISO 10646 combining characters should be treated):
- 1888
- How should collating sequences be defined for encoding schemes that represent single graphic symbols by combinations of codeset elements?
- 1889
- How are wildcard characters to work and how are numeric positions within strings to be defined for such encoding schemes?
- 1890
- 1891
- 1892
- (See conclusion (C-07)).
- 1893
- X/Open should work with bodies responsible for the standardisation of UNICODE, ISO-10646, the C programming language and SQL, to clarify how character strings are to be passed in SQL embedded variables or CLI function arguments between C programs and SQL implementations that use UNICODE or ISO 10646 (see conclusion (C-08)).
- 1894
- 1895
- 1896

1897 / *X/Open Technical Study*

1898 **Part 4**

1899 **X/Open DTP Specifications**

1900 *X/Open Company Ltd.*

Introduction

1901

1902 The chapters in this part of this technical study consider the impact of internationalisation on the
1903 X/Open Distributed Transaction Processing (DTP) specifications. These specifications are at
1904 present either snapshots, preliminary specifications or CAE specifications. They consist of the
1905 following documents (full details are given in **Referenced Documents** on page xv).

- 1906 • the **TX** (Transaction Demarcation) specification
- 1907 • the **XA** specification
- 1908 • the **XA+** specification.

1909 Structure of This Part

1910 Chapter 13 examines the implications of internationalisation on the DTP specifications listed
1911 above.

1912 Chapter 14 presents conclusions and recommendations.

1913 After this, Chapter 15 contains a set of *internationalisation* Change Requests (CRs) for each of the
1914 DTP specifications listed above. These are edited versions of standard X/Open Change
1915 Requests (CRs), in which the identity of the originator is omitted and the CRs are re-numbered
1916 into a sequential scheme, for the purposes of this document.

1917 **Note:** This version of this technical study does not consider the impact of internationalisation
1918 on the following X/Open DTP specifications:

- 1919 • the **TxRPC** specification
- 1920 • the **XATMI** specification
- 1921 • the **CPI-C** specification.

1922

1923 13.1 The TX (Transaction Demarcation) Specification**1924 13.1.1 Overview**

1925 The TX Specification provides an Application Program (AP) with an Application Programming
 1926 Interface (API) by which the AP can coordinate global transaction management with a
 1927 Transaction Manager (TM).

1928 The TX specification provides application programmers with an API in the following languages:

- 1929 • ISO C or Common Usage C
- 1930 • X/Open COBOL.

1931 These interfaces are functionally identical.

1932 For full details of this interface, see the referenced TX (Transaction Demarcation) specification.

1933 13.1.2 Internationalisation Implications**1934 Function Names, Arguments, Characteristics and Return Codes**

1935 In both C and Cobol, the TX function names, arguments and return codes have an English
 1936 language flavour. For example, to instruct the TM about transaction timeout information, the
 1937 AP uses the *tx_set_transaction_timeout()* function in C, or the TXSETTIMEOUT function in
 1938 COBOL.

1939 Similarly, the timeout value is specified in *timeout* (or **TRANSACTION-TIMEOUT**, and the
 1940 function ultimately returns [TX_OK] on successful completion.

1941 The TX interface is therefore convenient for English-speaking application programmers but not
 1942 for those of other nationalities. (The end user is not affected because the TX interface is not
 1943 directly visible at run time.)

1944 The current version of the X/Open Internationalisation Guide only discusses
 1945 internationalisation in terms of the end user. It focuses on providing internationalised
 1946 applications that can modify their behaviour at run time for specific language operation. At the
 1947 present time, there is no basis for internationalising the names of functions, arguments and
 1948 return codes such as those provided by the TX interface. Changes to this aspect of the TX
 1949 interface are therefore beyond the scope of this technical study.

1950 **The <tx.h> Header**

1951 The <tx.h> header defines a public structure called an **XID** to identify a transaction branch. The
 1952 contents of **XID** are used between all components that take part in a global transaction, within or
 1953 across TM domains.

1954 The **XID** structure is specified in the <tx.h> header as follows:

```

1955       #define XIDDATASIZE 128           /* size in bytes */
1956       struct xid_t {
1957           long formatID;               /* format identifier */
1958           long gtrid_length;          /* value not to exceed 64 */
1959           long bqual_length;         /* value not to exceed 64 */
1960           char data[XIDDATASIZE]; /* may contain binary data */
1961       };
1962       typedef struct xid_t XID;
1963       /*
1964        * A value of -1 in formatID means that the XID is null.
1965        */

```

1966 Although the field *data* is of type **char** and might, at first sight, be a candidate for conversion to
 1967 type **wchar_t** to allow for multi-byte character encodings, this is not necessary because its
 1968 significant length at any moment is defined by *gtrid_length* plus *bqual_length*. It does not rely on
 1969 being null terminated.

1970 However, Section 4.2 of the **TX** specification states that “APs may use **XIDs** for administrative
 1971 purposes such as auditing and logging”. It then warns that “the AP should treat each
 1972 component of *data* as an arbitrary collection of octets because, for instance, a component may
 1973 contain binary data as well as printable text”.

1974 Section 4.2 should contain an additional warning for APs that use **XIDs** for administrative
 1975 purposes such as auditing and logging. Because an **XID** may be encoded using a different,
 1976 possibly multi-byte, character set to the one specified by the current *locale*, the AP should take
 1977 care only to record (for these additional purposes) the contents of **XID** as the exact sequence of
 1978 bits in which it was received. The AP should not rely on being able to interpret these bits as
 1979 printable characters and should certainly avoid trying to display the value of any **XID** to an end
 1980 user at run time.

1981 13.2 The XA Specification

1982 13.2.1 Overview

1983 The XA interface is the bidirectional interface between a Transaction Manager (TM) and a
 1984 Resource Manager (RM). It lets a TM structure the work of RMs into global transactions and
 1985 coordinate transaction completion or recovery.

1986 The XA specification provides an API in the following languages:

- 1987 • ISO C or Common Usage C.

1988 For full details of this interface, see the referenced XA specification.

1989 13.2.2 Internationalisation Implications

1990 Function Names, Arguments, Characteristics and Return Codes

1991 In the same way as already described for the TX (Transaction Demarcation) specification (see
 1992 Section 13.1.2 on page 85 of this technical study), the XA function names, arguments and return
 1993 codes have an English language flavour, and are therefore convenient for English-speaking
 1994 software developers but not for those of other nationalities. (The XA interface is not directly
 1995 visible to either the application programmer or the end user.)

1996 Changes to this aspect of the XA specification are therefore beyond the scope of this document.

1997 The <xa.h> Header

1998 The <xa.h> header defines a public structure called an **XID** to identify a transaction branch. The
 1999 contents of XID are used between all components that take part in a global transaction, within or
 2000 across TM domains.

2001 The **XID** structure is specified in the <xa.h> header as follows:

```

2002     #define XIDDATASIZE  128          /* size in bytes */
2003     #define MAXGTRIDSIZE 64          /* maximum size in bytes of gtrid */
2004     #define MAXBQUALSIZE 64          /* maximum size in bytes of bqual */
2005     struct xid_t {
2006         long formatID;              /* format identifier */
2007         long gtrid_length;          /* value 1-64 */
2008         long bqual_length;          /* value 1-64 */
2009         char data[XIDDATASIZE];
2010     };
2011     typedef struct xid_t XID;
2012     /*
2013     * A value of -1 in formatID means that the XID is null.
2014     */
  
```

2015 Although the field *data* is of type **char** and might, at first sight, be a candidate for conversion to
 2016 type **wchar_t** to allow for mult-byte character encodings, this is not necessary because its
 2017 significant length at any moment is defined by *gtrid_length* plus *bqual_length*. It does not rely on
 2018 being null terminated.

2019 However, unlike the TX (Transaction Demarcation) specification, the XA specification does not
 2020 explicitly highlight that the field *data* should be treated as an arbitrary collection of octets that
 2021 might contain binary data.

2022 Resource Manager Name

2023 The Resource Manager Switch (**xa_switch_t**) includes the field *name*[*RMNAMESZ*] to contain the
2024 name of the resource manager. This field is of type **char** and has a defined length of 32
2025 characters including the null terminator.

2026 The **XA** specification does not say why this field is both fixed length *and* null terminated. The
2027 specification also does not define the purpose of this field and the possible ways that it might be
2028 used at run time by the different DTP components.

2029 There seem to be two alternative ways of making this field suitable for internationalisation:
2030 either

2031 • drop the null terminator, leave the field as a fixed length of 32 characters of type **char**, and
2032 specify that all characters must be initialised

2033 or

2034 • retain the null terminator and define that the field has a maximum length of 32 characters of
2035 type **wchar_t**, including the null terminator.

2036 In either case, because the field may be initialised with multi-byte characters encodings, the **XA**
2037 specification should warn that the field should be treated as an arbitrary collection of characters
2038 that may have been encoded using a different, possibly multi-byte, encoding scheme to the one
2039 currently defined for the present *locale*.

2040 In this technical study, the related Change Request (CR XA/I18N-02) specified in Section 15.2 on
2041 page 95 takes the second of these two options.

2042 XA Information String

2043 The Transaction Manager (TM) uses the functions *xa_open()* and *xa_close()* respectively to open
2044 and close a Resource Manager (RM). Both functions have an argument, *xa_info*, which points to
2045 a null-terminated character string that may contain instance-specific information for the RM.
2046 This field requires conversion to type **wchar_t** to allow for information strings that are encoded
2047 using multi-byte characters.

2048 13.3 The XA+ Specification

2049 13.3.1 Overview

2050 The XA+ interface provides an interface between a Transaction Manager (TM) and a
2051 Communication Resource Manager (CRM) to allow global transaction information to flow
2052 across TM domains. It also includes the XA interface described in Section 13.2 on page 87 of this
2053 technical study.

2054 The XA+ specification provides an API in the following languages:

- 2055 • ISO C or Common Usage C.

2056 For full details of this interface, see the referenced XA+ specification.

2057 13.3.2 Internationalisation Implications

2058 Function Names, Arguments, Characteristics and Return Codes

2059 In the same way as already described for the TX (Transaction Demarcation) specification (see
2060 Section 13.1.2 on page 85 of this technical study), the XA+ function names, arguments and return
2061 codes have an English language flavour, and are therefore convenient for English-speaking
2062 software developers but not for those of other nationalities. (The XA+ interface is not directly
2063 visible to either the application programmer or the end user.)

2064 Changes to this aspect of the XA+ specification are therefore beyond the scope of this document.

2065 The <xa.h> Header

2066 In the same way as already described for the XA specification (see Section 13.2.2 on page 87 of
2067 this technical study), The <xa.h> header in the XA+ specification defines a public structure
2068 called an **XID** to identify a transaction branch.

2069 Like XA, the XA+ specification does not explicitly highlight that the field *data*, which contains
2070 the global transaction identifier *gtrid* plus branch qualifier *bqual*, should be treated as an arbitrary
2071 collection of octets that might contain binary data.

2072 Resource Manager Name

2073 In the same way as already described for the XA specification (see Section 13.2.2 on page 87 of
2074 this technical study), the Resource Manager Switch (**xa_switch_t**) includes the field
2075 **name[RMNAMESZ]** to contain the name of the resource manager. Like XA, the XA+
2076 specification defines this field as type **char** with a defined length of 32 characters including the
2077 null terminator.

2078 The XA+ specification does not say why this field is both fixed length *and* null terminated.
2079 The specification also does not define the purpose of this field and the possible ways that it
2080 might be tested at run time by the different DTP components.

2081 Like XA, there are two alternative ways of making this field suitable for internationalisation:
2082 either

- 2083 • drop the null terminator, leave the field as a fixed length of 32 characters of type **char**, and
2084 specify that all characters must be initialised

2085 or

2086 • retain the null terminator and define that the field has a maximum length of 32 characters of
2087 type **wchar_t**, including the null terminator.

2088 In either case, because the field may be initialised with multi-byte characters encodings, the XA+
2089 specification should warn that the field should be treated as an arbitrary collection of characters
2090 that may have been encoded using a different, possibly multi-byte, encoding scheme to the one
2091 currently defined for the present *locale*.

2092 In this technical study, the related Change Request (CR XA+/I18N-02) specified in Section 15.3
2093 on page 98 takes the second of these two options.

2094 **XA Information String**

2095 In the same way as already described for the XA specification (see Section 13.2.2 on page 87), the
2096 functions that the TM uses to open and close an RM, *xa_open()* and *xa_close()* respectively, have
2097 an argument *xa_info* which points to a null-terminated character string that may contain
2098 instance-specific information for the RM. This field requires conversion to type **wchar_t** to allow
2099 for information strings that are encoded using multi-byte characters.

2100 **Blob Data**

2101 A Communication Resource Manager (CRM) can use the function *ax_set_branch_info()* to save
2102 information about a transaction branch. The CRM can later access this information using the
2103 function *ax_get_branch_info()*.

2104 The *blob* argument points to the character string of information to be saved. This argument is of
2105 type **char**. Although this field might, at first sight, be a candidate for conversion to type **wchar_t**
2106 to allow for multi-byte character encodings, this is not necessary because, in each instance of its
2107 use, its length is defined by the argument *blob_len*. Its length does not rely on null termination.

2108 However, the XA+ specification does not explicitly state that *blob* may contain characters
2109 encoded using a different character set to the one being used in the current *locale*, and it does not
2110 explicitly highlight that *blob* should be treated as an arbitrary collection of characters that might
2111 contain binary data.

2112 The XA+ specification should also warn implementors that the only valid method of
2113 determining the length of *blob* is by reference to *blob_len* and that any reliance on null
2114 termination may give unreliable results.

Conclusions and Recommendations

2115

2116 This chapter summarises the implications of internationalisation requirements on X/Open
 2117 Distributed Transaction Processing (DTP) specifications and presents conclusions and
 2118 recommendations.

2119 14.1 Summary

2120 In general, there are few problems in using the X/Open DTP specifications internationally.
 2121 None of the DTP specifications examined are directly concerned with the control or display of
 2122 data that is used or manipulated by the end user, and are therefore not directly impacted by the
 2123 internationalisation requirements of different national languages and cultural conventions.

2124 The changes that are required relate mainly to:

- 2125 • converting to wide characters (**wchar_t**) those character strings that could be misinterpreted
 2126 by different DTP components using different, and possibly multi-byte, encoding methods
- 2127 • inserting clarifications where data stored by one DTP component in an apparently character
 2128 format should only be treated by connected DTP components as arbitrary collections of
 2129 binary data.

2130 The changes that this technical study recommends are shown as a set of X/Open Change
 2131 Requests (CRs) in Chapter 15.

2132 14.2 Function Names, Arguments, Characteristics and Return Codes

2133 As described in Section 13.1.2 on page 85, the function names, arguments and return codes used
 2134 in the DTP specifications have an English language flavour. For example, to set transaction
 2135 timeout information in the Transaction Manager (TM), the AP uses the function
 2136 *tx_set_transaction_timeout()*.

2137 The DTP interfaces are therefore convenient for English-speaking application programmers and
 2138 software developers but not for those of other nationalities.

2139 The current version of the X/Open **Internationalisation Guide** only discusses
 2140 internationalisation in terms of the end user, and changes in this area are therefore beyond the
 2141 scope of this technical study.

2142 **14.3 ISO C and Common Usage C**

2143 Chapter 4 of the **XA** and **XA+** specifications states that the `<xa.h>` header file is suitable for both
2144 ISO C and Common Usage C implementations.

2145 Chapter 15 of this technical study includes Change Requests (CRs) that specify changing certain
2146 fields of type `char` into the wide character type `wchar_t`. Although ISO C supports the `typedef`
2147 name `wchar_t`, it not certain that all implementations of Common Usage C support it also. If
2148 they do not, then the statement in **XA** and **XA+** that the `<xa.h>` header file is suitable for both
2149 ISO C and Common Usage C becomes invalid.

2150 This document does not include CRs that address this point. However, if not all Common
2151 Usage C implementations support `wchar_t`, then the **XA** and **XA+** specifications should either
2152 remove the statement about Common Usage C, or should qualify the statement by saying: "...
2153 suitable for ISO C and for Common Usage C implementations that support the `typedef` name
2154 `wchar_t`".

2155 The problem does not affect the **TX** (Transaction Demarcation) specification because no changes
2156 to use `wchar_t` are proposed.

Change Requests for Internationalisation

2157

2158

This chapter contains formal change requests for the X/Open distributed transaction processing specifications.

2159

2160

The X/Open Transaction Processing Working Group is currently evaluating these change requests.

2161

2162 **15.1 The TX (Transaction Demarcation) Specification**

2163 Document: The TX (Transaction Demarcation) Specification
2164 X/Open Preliminary Specification, P209 (October 1992).

2165 Change number: TX/I18N-01

2166 Title: Unique Transaction Identifier (**XID**)

2167 Qualifier: Minor Technical

2168 Rationale: An Application Program (AP) uses *tx_info()* to obtain **XID** information to
2169 identify in which global transaction it is currently located. After the
2170 Transaction Manager (TM) has returned this information, the AP may use it
2171 for its own administrative purposes, such as auditing and logging.

2172 Because the *data* component contained within the **XID** may have been
2173 encoded using a different, possibly multi-byte, character set to the one
2174 specified by the current *locale*, the AP should take care only to record (for
2175 such purposes) the contents of *data* as the exact sequence of bits in which it
2176 was received. The AP should not rely on being able to interpret these bits as
2177 printable characters and should not attempt to display the value of any **XID**
2178 to an end user at run time.

2179 Change: In Section 4.2, at the end of the paragraph that currently says:

2180 “An important attribute of the **XID** is global uniqueness, based on the exact
2181 order of the bits in the *data* element of the **XID** for the lengths specified. The
2182 AP should treat each component of *data* as an arbitrary collection of octets
2183 because, for instance, a component may contain binary data as well as
2184 printable text.”

2185 Add the following text:

2186 “Because the *data* component contained within the **XID** may have been
2187 encoded using a different, possibly multi-byte, character set to the one
2188 specified by the current *locale*, the AP should take care only to record the
2189 contents of *data* as the exact sequence of bits in which it was received. The
2190 AP should not rely on being able to interpret these bits as printable
2191 characters and should not attempt to display the value of any **XID** to an end
2192 user at run time.”

2193 **15.2 The XA Specification**

2194	Document:	The XA Specification
2195		X/Open CAE Specification, C193 (October 1991).
2196	Change number:	XA/I18N-01
2197	Title:	Unique Transaction Identifier (XID)
2198	Qualifier:	Minor Technical
2199	Rationale:	The < xa.h > header defines a public structure called an XID to identify a transaction branch. The contents of XID are used between all components that take part in a global transaction, within or across TM domains. Unlike the TX (Transaction Demarcation) specification, the XA specification does not explicitly warn that each component of the <i>data</i> portion of the XID should be treated as a string of bits rather than as recognisable characters.
2200		
2201		
2202		
2203		
2204		
2205	Change:	In Section 4.2, at the end of the paragraph that currently says:
2206		“Although “ xa.h ” constrains the length and byte alignment of ... that the XID is null.”
2207		
2208		Add the following text:
2209		“An important attribute of the XID is global uniqueness, based on the exact order of the bits in the <i>data</i> element of the XID for the lengths specified. Each component of <i>data</i> should be treated as an arbitrary collection of octets because, for instance, a component may contain binary data as well as printable text, and it may have been encoded using a different, and possibly multi-byte, encoding scheme to the one active for the current <i>locale</i> .”
2210		
2211		
2212		
2213		
2214		

- 2215 Document: The XA Specification
 2216 X/Open CAE Specification, C193 (October 1991).
- 2217 Change number: XA/I18N-02
- 2218 Title: Internationalised Resource Manager Name
- 2219 Qualifier: Minor Technical
- 2220 Rationale: The Resource Manager Switch (`xa_switch_t`) includes the field
 2221 `name[RMNAMESZ]` to contain the name of the resource manager. This field
 2222 is of type `char` and is null terminated. This field requires conversion to type
 2223 `wchar_t` to allow for Resource Managers (RMs) that define their names using
 2224 multi-byte character encodings.
- 2225 Change:
- 2226 i. In the second paragraph of Section 4.3, add the following text:
- 2227 “The RM name is a field of type `wchar_t` that may contain characters
 2228 from the character set of any natural language, encoded using any
 2229 encoding scheme. The TM should not rely on this field being encoded
 2230 in any particular encoding scheme and should treat its contents only as
 2231 an arbitrary collection of characters. The TM should not display the
 2232 field in the form of printable characters to users (who may be working
 2233 in a different *locale* to the one in which the field was originally
 2234 encoded).”
- 2235 ii. Add the above text also at the end of the second dash item of **Public**
 2236 **Information** in section 7.2.
- 2237 iii. In the switch structure in Section 4.3, change:
- 2238 `char name[RMNAMESZ];`
 2239 `/* name of resource manager */`
- 2240 to
- 2241 `wchar_t name[RMNAMESZ];`
 2242 `/* name of resource manager */`
- 2243 iv. Make the above change also in the equivalent section of Appendix A.

2244 Document: The XA Specification
 2245 X/Open CAE Specification, C193 (October 1991).

2246 Change number: XA/I18N-03

2247 Title: XA Information Strings

2248 Qualifier: Minor Technical

2249 Rationale: The functions that the TM uses to open and close an RM, *xa_open()* and
 2250 *xa_close()* respectively, have an argument *xa_info* which points to a null-
 2251 terminated character string that may contain instance-specific information
 2252 for the RM. This field requires conversion to type **wchar_t** to allow for
 2253 information strings that are encoded using multi-byte characters.

2254 Change:

2255 i. In Section 4.3, change:

```
2256         int (*xa_open_entry)(char *, int, long);
2257             /* xa_open function pointer */
2258         int (*xa_close_entry)(char *, int, long);
2259             /* xa_close function pointer */
```

2260 to:

```
2261         int (*xa_open_entry)(wchar_t *, int, long);
2262             /* xa_open function pointer */
2263         int (*xa_close_entry)(wchar_t *, int, long);
2264             /* xa_close function pointer */
```

2265 ii. Make the above change also in the equivalent section of Appendix A.

2266 iii. Make the equivalent change in the **SYNOPSIS** section in the manual
 2267 page for *xa_close()* in Chapter 5.

2268 iv. Make the equivalent change in the **SYNOPSIS** section in the manual
 2269 page for *xa_open()* in Chapter 5.

2270 **15.3 The XA+ Specification**

2271	Document:	The XA+ Specification
2272		X/Open Snapshot, Version 2, S423 (July 1994).
2273	Change number:	XA+/I18N-01
2274	Title:	Unique Transaction Identifier (XID)
2275	Qualifier:	Minor Technical
2276	Rationale:	The <xa.h> header defines a public structure called an XID to identify a transaction branch. The contents of XID are used between all components that take part in a global transaction, within or across TM domains. Unlike the TX (Transaction Demarcation) specification, the XA+ specification does not explicitly warn that each component of the <i>data</i> portion of the XID should be treated as a string of bits rather than as recognisable characters.
2277		
2278		
2279		
2280		
2281		
2282	Change:	In Section 4.2, at the end of the paragraph that currently says:
2283		“Although <xa.h> constrains the length and byte-alignment of ... that the XID is null.”
2284		
2285		Add the following text:
2286		“An important attribute of the XID is global uniqueness, based on the exact order of the bits in the <i>data</i> element of the XID for the lengths specified. Each component of <i>data</i> should be treated as an arbitrary collection of octets because, for instance, a component may contain binary data as well as printable text, and it may have been encoded using a different, and possibly multi-byte, encoding scheme to the one active for the current <i>locale</i> .”
2287		
2288		
2289		
2290		
2291		

2292 Document: The XA+ Specification
 2293 X/Open Snapshot, Version 2, S423 (July 1994).

2294 Change number: XA+/I18N-02

2295 Title: Internationalised Resource Manager Name

2296 Qualifier: Minor Technical

2297 Rationale: The Resource Manager Switch (`xa_switch_t`) includes the field
 2298 `name[RMNAMESZ]` to contain the name of the resource manager. This field
 2299 is of type `char` and is null terminated. This field requires conversion to type
 2300 `wchar_t` to allow for Resource Managers (RMs) that define their names using
 2301 multi-byte character encodings.

2302 Change:

2303 i. In the second paragraph of Section 4.4, add the following text:

2304 “The RM name is a field of type `wchar_t` that may contain characters
 2305 from the character set of any natural language, encoded using any
 2306 encoding scheme. The TM should not rely on this field being encoded
 2307 in any particular encoding scheme and should treat its contents only as
 2308 an arbitrary collection of characters. The TM should not display the
 2309 field in the form of printable characters to users (who may be working
 2310 in a different *locale* to the one in which the field was originally
 2311 encoded).”

2312 ii. Add the above text also at the end of the second dash item of **Public**
 2313 **Information** in section 7.2.

2314 iii. In the switch structure in Section 4.4, change:

```
2315 char name[RMNAMESZ];
2316         /* name of resource manager */
```

2317 to

```
2318     wchar_t name[RMNAMESZ];
2319         /* name of resource manager */
```

2320 iv. Make the above change also in the equivalent section of Appendix A.

2321 Document: The XA+ Specification
 2322 X/Open Snapshot, Version 2, S423 (July 1994).

2323 Change number: XA+/I18N-03

2324 Title: XA Information Strings

2325 Qualifier: Minor Technical

2326 Rationale: The functions that the TM uses to open and close an RM, *xa_open()* and
 2327 *xa_close()* respectively, have an argument *xa_info* which points to a null-
 2328 terminated character string that may contain instance-specific information
 2329 for the RM. This field requires conversion to type **wchar_t** to allow for
 2330 information strings that are encoded using multi-byte characters.

2331 Change:

2332 i. In Section 4.4, change:

```
2333         int (*xa_open_entry)(char *, int, long);
2334             /* xa_open function pointer */
2335         int (*xa_close_entry)(char *, int, long);
2336             /* xa_close function pointer */
```

2337 to:

```
2338         int (*xa_open_entry)(wchar_t *, int, long);
2339             /* xa_open function pointer */
2340         int (*xa_close_entry)(wchar_t *, int, long);
2341             /* xa_close function pointer */
```

2342 ii. Make the above change also in the equivalent section of Appendix A.

2343 iii. Make the equivalent change in the **SYNOPSIS** section in the manual
 2344 page for *xa_close()* in Chapter 5.

2345 iv. Make the equivalent change in the **SYNOPSIS** section in the manual
 2346 page for *xa_open()* in Chapter 5.

2347	Document:	The XA+ Specification
2348		X/Open Snapshot, Version 2, S423 (July 1994).
2349	Change number:	XA+/I18N-04
2350	Title:	Transaction Branch Information (<i>blob</i> data)
2351	Qualifier:	Minor Technical
2352	Rationale:	A Communication Resource Manager (CRM) can use the function
2353		<i>ax_set_branch_info()</i> to save information about a transaction branch. The
2354		CRM can later access this information using the function
2355		<i>ax_get_branch_info()</i> .
2356		The <i>blob</i> argument points to the character string of information to be saved.
2357		This argument is of type char . Its length is defined by the argument <i>blob_len</i> .
2358		Its does not rely on being null terminated.
2359		However, the XA+ specification does not explicitly state that <i>blob</i> may
2360		contain characters encoded using a different character set to the one being
2361		used in the current <i>locale</i> , and it does not explicitly highlight that <i>blob</i> should
2362		be treated as an arbitrary collection of characters that might contain binary
2363		data.
2364		The XA+ specification should also warn implementors that the only valid
2365		method of determining the length of <i>blob</i> is by reference to <i>blob_len</i> and that
2366		any reliance on null termination may give unreliable results.
2367	Change:	
2368		i. In the manual page for <i>ax_get_branch_info()</i> in Chapter 5, after the
2369		paragraph that reads:
2370		“The <i>blob_len</i> argument is a pointer to an area in which the transaction
2371		manager returns the size of <i>blob</i> .”
2372		Add the following new paragraph:
2373		“The <i>blob</i> argument may contain characters encoded using a different
2374		character set to the one being used in the current <i>locale</i> . It should
2375		therefore be treated as an arbitrary collection of characters that might
2376		contain binary data. The only valid method of determining the length
2377		of <i>blob</i> is by reference to <i>blob_len</i> . Any reliance on null termination may
2378		give unreliable results.”
2379		ii. In the manual page for <i>ax_set_branch_info()</i> in Chapter 5, add the same
2380		new paragraph as described above.

2381 / *X/Open Technical Study*

2382 **Part 5**

2383 **X/Open Systems Management Specifications**

2384 *X/Open Company Ltd.*

Introduction

2385

2386 Systems management applications are among those most likely to be affected by
2387 internationalisation issues. This is because management of a distributed, multi-national system
2388 implies management of resources owned or used by all of the users of the system. These users
2389 are likely to work in diverse language and cultural environments. As an added complication,
2390 there may be several systems managers, who may themselves work in diverse language and
2391 cultural environments. The systems management components must be designed to support this.

2392 The chapters in this part of this technical study consider the impact of internationalisation on the
2393 X/Open systems management specifications. These specifications are at present either
2394 snapshots, preliminary specifications or CAE specifications.

2395 They consist of the following documents (full details are given in **Referenced Documents** on
2396 page xv).

- 2397 • the **XMP** specification
- 2398 • the **XMPP** specification
- 2399 • the **XGDMO** specification
- 2400 • the Performance Management specification (contained in the **UMA** guide, the **UMA DCI**
2401 specification, the **UMA MLI** specification and the **UMA DPD** specification)
- 2402 • the **XBSA** specification
- 2403 • the **XSMS** specification.

2404 **Structure of This Part**

2405 Chapter 17 examines the implications of internationalisation on the systems management
2406 specifications listed above.

2407 Chapter 18 presents conclusions and recommendations.

2408 After this, Chapter 19 contains a set of *internationalisation* Change Requests (CRs) for each of the
2409 systems management specifications listed above. These are edited versions of standard X/Open
2410 Change Requests (CRs), in which the identity of the originator is omitted and the CRs are re-
2411 numbered into a sequential scheme, for the purposes of this document.

2412

2413 17.1 The XMP Specification**2414 17.1.1 Overview**

2415 The **XMP** specification defines an Application Programming Interface (API) to management
 2416 information services. It can be used in conjunction with the OSI systems management protocol
 2417 defined in the CMISP standard or with the Internet systems management protocol defined in the
 2418 SNMP Internet RFC. Its use in conjunction with other protocols is not precluded, but is not
 2419 specified by the **XMP** specification.

2420 For full details of this interface, see the referenced **XMP** specification.

2421 17.1.2 Internationalisation Implications**2422 Error Messages**

2423 The **Error-Message** function returns a text string that describes an error. Its description refers to
 2424 the “X/Open Native Language System (NLS)”, but does not specify how use of the NLS affects
 2425 the string returned, or constrain the character set that may be used. The length of the string is
 2426 given explicitly, and the string is null-terminated. Presumably, this means that it is terminated
 2427 by a single null element of type char; this means that encodings such as ISO 10646 or UNICODE
 2428 that have character representations containing embedded nulls would cause applications that
 2429 use the null terminator to behave incorrectly.

2430 Use of Latin Alphabet Strings

2431 The specification of class **Entity-Name** defines attribute **entity** as a printable string. This means
 2432 that management application names and system names are essentially restricted to using ASCII
 2433 characters.

2434 **Note:** The specification of class **Name-String** defines attribute **name-String** as an IA5 string.
 2435 This does not mean that names are restricted to using the characters of International
 2436 Alphabet nr. 5 (which is a subset of the ASCII character set), since a name can be of
 2437 class **DS-DN** or **SNMP-Object-Name**, and these are not restricted to particular
 2438 character sets.)

2439 String Comparisons

2440 The specification of class **Filter-Item** defines attribute **substrings** whose meaning in an
 2441 internationalised context is unclear. For example:

- 2442 • the interpretation of “Initial Substring” and “Final Substring” for languages with mixed
 2443 directionality requires some thought
- 2444 • what constitutes a substring is open to question for languages in which a single character can
 2445 have different representations (for example, <e-acute>/<e>+<acute accent>)?

2446 Similarly, the meanings in an internationalised context of attributes **greater-or-equal** and **less-**
2447 **or-equal** of class **Filter-Item** are unclear (what collating order is assumed?)

2448 17.2 The XMPP Specification**2449 17.2.1 Overview**

2450 A protocol profile is the specification of a set of communication protocols, and of options within
2451 those protocols, to be used by communicating systems for a particular purpose. The **XMPP**
2452 specification defines the protocol profiles to be used for systems management within the
2453 X/Open CAE. The profiles include OSI communication protocols, Internet communication
2454 protocols, and some proprietary communication protocols. The **XMPP** specification defines the
2455 profiles by referring to OSI and Internet protocol and profile specifications.

2456 For full details of this interface, see the referenced **XMPP** specification.

2457 17.2.2 Internationalisation Implications

2458 There are no internationalisation implications for the **XMPP** specification.

2459 17.3 The XGDMO Specification

2460 17.3.1 Overview

2461 Managed objects are abstract representations of parts of computer systems and networks. They
2462 are used by systems management applications. Different managed objects are defined for
2463 different types of computer systems and networks but, because they have a common format,
2464 they can be handled by generic systems management software.

2465 The referenced GDMO standard describes how managed objects should be defined. It specifies
2466 templates into which the information for each managed object can be inserted in order to
2467 produce a formal definition of it. Although they are formal enough that they can be processed
2468 automatically, these descriptions are intended to be read by people.

2469 In the X/Open CAE, management applications can use the API defined in the **XMP** specification
2470 to invoke management communications services. In doing so, they will use the API defined in
2471 the **XOM** specification to manipulate the complex information structures that are
2472 communicated. The implementation of the **XMP** specification can include specific packages that
2473 support particular managed objects. Alternatively, it can provide a configuration utility that
2474 enables the required packages to be generated at compile time from a formalised description of
2475 the managed objects that they must support.

2476 The **XGDMO** specification describes how such a formalised description can be generated
2477 mechanically from the templates defined in the GDMO standard. The algorithm described also
2478 produces the header files that an application program written in the C programming language
2479 will need in order to use the XOM API to manipulate the information structures that represent
2480 the managed objects, and generates documentation that describes the packages that it has
2481 generated.

2482 For full details of this interface, see the referenced **XGDMO** specification.

2483 17.3.2 Internationalisation Implications

2484 The **XGDMO** specification in effect defines a language translator, and similar issues arise as for
2485 a programming language. In particular, there is the question of what character sets can be used
2486 in the input to the translator, and of what character sets can appear in its output. This question
2487 is not addressed in the **XGDMO** specification.

2488 It is implicitly assumed that the input character set includes the distinct lowercase and
2489 uppercase versions of the basic Latin alphabet, the decimal digits, and spacing and punctuation
2490 characters. Generated output includes specific characters (as in the "OMP_O-" prefix). OM
2491 classes and attributes are identified and alphabetised in ascending order.

2492 Lack of support for characters outside the basic Latin alphabet would mean that the natural
2493 names could not be used for managed objects and attributes that have been defined and named
2494 in a language and cultural environment other than an English one. This would affect the
2495 applications programmer, but should not affect the user of the applications programs.

2496 17.4 The UMA Specifications

2497 17.4.1 Overview

2498 The X/Open Universal Measurement Architecture (UMA) supports the collection, management
2499 and reporting of performance data and events. It defines four layers of functionality:

2500 **Measurement Application Layer**

2501 This consists of the various Measurement Application Programs (MAPs) that provide
2502 services for technical support of management goals. Examples of MAPs are performance
2503 monitors, capacity planning tools, and tuning advisors.

2504 **Data Services Layer**

2505 This accepts measurement requests from MAPs and supplies measurement data to the
2506 MAPs or to other destinations requested by them. Other destinations can include private
2507 files or a facility for access and maintenance of historical data known as UMA Data Storage.

2508 **Measurement Control Layer**

2509 This schedules and synchronises data collection and supplies the collected data to the Data
2510 Services Layer.

2511 **Data Capture Layer**

2512 This layer is responsible for collecting raw data and supplying it to the Measurement
2513 Control Layer.

2514 X/Open has produced the following UMA specifications:

- 2515 • the **UMA** guide, which provides an overview of the UMA
- 2516 • the **UMA MLI** specification, which defines:
 - 2517 — the interface through which the Measurement Application Layer invokes the services of
2518 the Data Services Layer
 - 2519 — the protocol used by implementations of the Data Services Layer in different machines to
2520 communicate with each other
- 2521 • the **UMA DCI** specification, which defines the interface through which the Measurement
2522 Control Layer invokes the services of the Data Capture Layer
- 2523 • the **UMA DPD** specification, which defines the format of data passed across the
2524 Measurement Layer Interface, and which may also be used internally within the Data
2525 Services Layer.

2526 **Note:** The interface between the Data Services and Measurement Control layers is not
2527 specified.

2528 For full details of these interfaces, see the referenced **UMA** specifications.

2529 **17.4.2 Internationalisation Implications**2530 **Use of Character Strings in the Data Pool**

2531 A number of data pool message fields defined in the **UMA DPD** specification consist of
2532 character strings. They include, for example:

- 2533 • the class name, subclass name and subclass abbreviated name from the Names subclass of
2534 the Configuration class
- 2535 • the command name from the Remote Terminal Monitor Measures subclass of the Response
2536 Time class
- 2537 • the partition name from the Disk Partition Data subclass of the Disk Device Data class.

2538 It is not clear what will happen:

- 2539 • when messages containing such data are processed by an internationalised application that
2540 uses a character set and encoding that is not compatible with those used by the Data Services
2541 Layer
- 2542 • when messages are passed between implementations of the Data Services Layer in different
2543 machines that use incompatible character sets and encodings.

2544 The strings are null-terminated, so the use of encodings such as ISO 10646 or UNICODE that
2545 include nulls in character encodings is problematic. However, the sizes of the text fields are
2546 given in the count that is provided in the Text Descriptor or the size that is provided in the Array
2547 Descriptor. The application therefore need not rely on the null terminator when determining the
2548 lengths of such fields.

2549 It is probable that the underlying assumption is that all applications will use character sets and
2550 encodings that are compatible with basic ASCII, and that the character strings in the messages
2551 will only use basic ASCII. If this is so, it is probably true for most systems today, but it does
2552 restrict the internationalisation of systems in future.

2553 It is also probable that many of these strings will be the same as the ASCII text strings in labels
2554 used in the Data Capture Interface. This facilitates the development of measurement control
2555 layer programs, but not in an internationalised context. For example, what would happen if a
2556 machine in Japan with a filesystem partition named using Kanji characters was managed by a set
2557 of management applications running on a machine in which the only supported character set
2558 encoding was ASCII?

2559 **Use of Character Strings in the Logical Message Protocol**

2560 The Logical Message Protocol is defined in the **UMA MLI** specification for communications
2561 between implementations of the Data Services Layer in different machines. A number of the
2562 message fields defined for this protocol are text strings (for example, the source and destination
2563 fields of the *Create* message). It is not clear what character set encodings are to be used for these
2564 fields, and it is not clear what happens when messages are passed between machines that use
2565 different character set encodings.

2566 The strings are null-terminated, so the use of encodings such as ISO 10646 or UNICODE that
2567 include nulls in character encodings is problematic. The specification should either preclude the
2568 use of such encodings or advise applications to rely on the count that is provided in the Text
2569 Descriptor or the size that is provided in the Array Descriptor, rather than relying on the null
2570 terminator, when determining the lengths of such fields.

2571 **Note:** The **UMA MLI** specification does not explicitly list these descriptors in the way that the
2572 **UMA DPD** specification does, but section 6.3.5 (Variable Length Data) of the **UMA**
2573 **MLI** specification appears to imply that they are present, immediately preceding the
2574 fields that they describe.

2575 It is probable that there is an implicit assumption that only basic ASCII encodings will be used.
2576 Again, this represents a restriction on the development of internationalised systems.

2577 **Textual Descriptions in UDU Control Segments**

2578 UMA API messages, called UMA Data Units (UDUs), are defined in the **UMA MLI** specification.
2579 They are passed across the Measurement Layer Interface. They can include control segments
2580 which, when passed from the Data Services Layer to a MAP, can contain status information.

2581 The bodies of such UDU control segments can contain textual descriptions of problems
2582 encountered. These are stated to be “useful for reporting the condition back to a user” but
2583 would not be usable by internationalised applications. It would be better to state that
2584 applications should use a message cataloguing mechanism to generate status messages.

2585 **String Arguments of Measurement Layer Interface Functions**

2586 A number of arguments of the functions defined in the **UMA MLI** specification are character
2587 strings. It is not clear what character set encodings can be used for these strings.

2588 For an internationalised application, it should be possible to use the character set encoding of the
2589 currently established locale.

2590 If this encoding allows a null byte to appear as part of the encoding of a character (as ISO 10646
2591 and UNICODE do), then the use of null terminators for strings becomes problematic. This
2592 problem can be avoided by specifying that the arguments are arrays of elements of type
2593 **wchar_t**.

2594 **String Data Capture Interface Data Types**

2595 Metric data objects passed across the Data Capture Interface described in the **UMA DCI**
2596 specification can be text strings. It is not clear what character set encodings can be used in such
2597 strings. It is also not clear what happens when the strings are passed between different
2598 machines that use different character set encodings.

2599 The strings are null-terminated, so the use of encodings such as ISO 10646 or UNICODE that
2600 include nulls in character encodings is problematic. The specification should either preclude the
2601 use of such encodings or advise applications to rely on the length field that is provided, rather
2602 than the null terminator, when determining the lengths of such fields.

2603 **Use of ASCII Strings in Data Capture Interface Labels**

2604 The **UMA DCI** specification states that it is a goal that any textual information should be
2605 capable of supporting an internationalised application. It defines structure type *DCILabel* which
2606 contains an ASCII text string and may also contain an internationalised label. The structure of
2607 such an internationalised label is unspecified, but it is suggested that it could identify a message
2608 catalogue and a message within that catalogue.

2609 Since the structure of an internationalised label is unspecified, it is not possible to write a
2610 portable application that uses it. However, the Data Capture Interface is not used by MAPs; it is
2611 used by programs of the Measurement Control Layer. In view of the way that strings are used in
2612 the Data Pool, in the Measurement Layer Interface, and elsewhere in the Data Capture Interface,
2613 it is unlikely that the availability of internationalised labels in the Data Capture Interface can

- 2614 very much affect the degree to which a MAP can be internationalised.
- 2615 The structure of internationalised labels could be specified, but it does not seem worthwhile to
- 2616 do this except as part of a general solution to the problem of string usage.

2617 17.5 The XBSA Specification

2618 17.5.1 Overview

2619 The **XBSA** Specification defines an API to services that can be used to back-up or to archive data,
2620 and to restore the data that has been backed up or archived. It is intended to be used by non-
2621 management applications that need back-up or archive services, and by management back-up
2622 and archive applications.

2623 For full details of this interface, see the referenced **XBSA** specification.

2624 17.5.2 Internationalisation Implications

2625 Each item of information that is backed-up or archived is created in a particular locale and, more
2626 importantly, resides in a file whose name is created in a particular locale. Different users who
2627 created the files may have names that assume different locales and may use different locales for
2628 their filenames.

2629 An internationalised application uses the locale established by the current user. In the case of
2630 the **XBSA** API, the current user could be a system administrator (in the case of a back-up or
2631 archive management application) or an ordinary user (in the case of a non-management
2632 application).

2633 The names of users and files are passed over the API in character strings. The strings can be
2634 lexicographically compared with each other, and these comparisons can use wildcards.

2635 There is no mention of the locale that is to be used to interpret these strings. (Does “Ålborg”
2636 come before “Amsterdam” or after “Zurich”?) There is no mention of how wildcards are to be
2637 interpreted in locales that include combining characters. (Does “e*” match “écoutez”?)

2638 Different names may have been encoded using different character set encodings. For example,
2639 one user might name files in EBCDIC while another uses UNICODE. This is more likely to occur
2640 in distributed system, in which different computers may have different *native* codesets and
2641 character encodings, but may also occur in a single computer that supports multiple locales.

2642 In addition to filenames and information-creator names, there are other attributes that are
2643 passed over the API as character strings (for example, domain names and policy set names).
2644 Also, rules and schedules are character strings. There is no mention of these strings being in the
2645 character set of the current locale.

2646 Although the **XBSA** specification does not say so, the variable length character strings that are
2647 passed across the API are presumably null-terminated. This would give problems if the
2648 characters are encoded in UNICODE or ISO 10646.

2649 There are also attributes that are date/time specifications in particular formats, for example
2650 ddmmyy/hh:mm:ss. This is stated in Section 2.8 (Object Descriptors and the BSA Catalog), but
2651 there is no description of the specific attributes concerned.

2652 17.6 The XSMS Specification

2653 17.6.1 Overview

2654 The XSMS specification presents a system administration framework that includes both object-
2655 oriented programming (based on the CORBA architecture and specification) and the X/Open
2656 distributed systems management reference model (described in the XRM specification).

2657 The framework consists of a set of services. They include both systems management services
2658 (such as the *policy management service*) and OMG object management⁷ services (such as the *object*
2659 *life cycle service*).

2660 The systems management services are specified using OMG Interface Definition Language
2661 (OMG IDL). The OMG object management services that are required for systems management
2662 are discussed, but are mostly not specified in detail. Specifications of these services either have
2663 already been created by the OMG, or are expected to be created by the OMG in the future.

2664 So that the services can be accessed from shell scripts, a set of type mappings and a set of
2665 commands are defined. These effectively specify a shell binding for the OMG IDL.

2666 For full details of this interface, see the referenced XSMS specification.

2667 17.6.2 Internationalisation Implications

2668 General

2669 While the XSMS specification states the importance of the requirement for internationalisation,
2670 it also states that internationalisation requirements for implementations are outside its scope.
2671 Internationalisation requirements have not been addressed in detail in the definition of the
2672 interfaces.

2673 Names

2674 Types of objects, and instances of those types, have human-readable names. Their purpose is to
2675 identify the object types and instances at the user interface. The XSMS specification does not
2676 define the user interface. The user interface would be implemented by application programs
2677 and shell scripts using the IDL interfaces that XSMS specification does define. In these
2678 interfaces, the names are represented as character strings.

2679 Other entities, including filters and actions, have similar names. These names are also
2680 represented by strings.

2681 The XSMS specification does not specify how these strings are encoded.

2682 There may be a need in some systems to make the names of object types depend on the current
2683 locale. For example, an English administrator might wish to use the term *computer* and a French
2684 one might wish to use *ordinateur* to refer to the same type of object. There could also be a need to
2685 use locale-dependent names for object instances, just as locale-dependent names may be used
2686 for files (see Section 17.5 on page 115).

2687 _____
2688 7. The concept of *object* as used by the OMG must be distinguished from the concept of *managed object*. OMG objects may be used as
2689 the programming constructs that represent managed objects.

2690 Null-Terminated Strings

2691 The description of the *filter* operation refers to the *value* argument being a null-terminated string.
2692 The operation is defined in IDL, and the representation of the string datatype is presumably a
2693 feature of the binding of IDL to the programming language (eg. C or C++) that is used. If this
2694 binding maps IDL strings to null-terminated strings, then use of UNICODE or ISO 10646
2695 character encodings becomes problematic.

2696 String Arguments to Commands

2697 In the command line interface, strings are typically delimited by white space characters (space,
2698 tab, newline etc.) or by quote or double-quote characters. The characters that can be used, and
2699 their encodings, depend on the currently established locale.

2700 String Comparisons

2701 The filter operations include string comparisons. The semantics of these operations should be
2702 locale-dependent. This is however not discussed in the **XSMS** specification.

2703 Exceptions

2704 The IDL data types used in handling exception conditions are defined in the SysAdminExcept
2705 module. They include strings giving resource names, operation names and default messages.
2706 The semantics of these strings are not described in the **XSMS** specification. They should be
2707 locale-dependent.

Conclusions and Recommendations

2708

2709 This chapter summarises the implications of internationalisation requirements on X/Open
2710 systems management specifications and presents conclusions and recommendations.

2711 18.1 Conclusions

2712 In a multi-national enterprise, the users of a system, and even the managers of a system, may
2713 work in a number of different language and cultural environments. Systems management
2714 applications must be able to support the management of such systems.

2715 This gives rise to a number of issues in the specifications covered by this part of this technical
2716 study. They are discussed in the following subsections.

2717 18.1.1 Character String Identifiers

2718 A number of entities - including managed object classes, managed object instances, systems, files
2719 and filters - are identified by character strings. In some cases, those strings are specified to
2720 consist of ASCII text. In other cases, the character set and encoding are not specified, but there is
2721 no provision for them to depend on the current locale.

2722 This issue affects

- 2723 • the UMA Data Pool, since character string identifiers can appear in data pool messages
- 2724 • the UMA logical message protocol, since character string identifiers can appear in fields of
2725 messages of this protocol
- 2726 • the UMA measurement layer interface, since character string identifiers can appear as
2727 function arguments in this interface
- 2728 • the UMA data capture interface, since labels can include text strings
- 2729 • the Backup/Restore interface, which uses character strings to name files, systems, domains,
2730 policy sets and other entities
- 2731 • the Management Services for an OMG Environment, since character strings are used for
2732 names in IDL operations (and in the mappings of those operations to the command line
2733 interface) and in exception descriptions.

2734 Use of character strings to identify entities creates problems when the user of the entity and the
2735 manager of the entity use different character sets and encodings. It may be difficult for the user,
2736 for the manager, or for both of them to work effectively. For example, the manager may be
2737 unable to restore a file from archive because his locale does not include some of the characters
2738 used in the filename.

2739 There are some cases where it would be wrong to use anything other than a character string. For
2740 example, user names are normally stored as character strings, and it is hard to envisage how else
2741 they could be stored. Thus, it is necessary to find a solution that enables character strings to be
2742 used in some cases, as opposed to a solution that replaces all character strings by some other
2743 form of identification.

2744 Where use of the Latin alphabet (as in IA5 strings or printable strings) is an option that is open to
 2745 the application programmer, but is not essential, the writing of internationalised applications is
 2746 not precluded. However, writers of applications who assume an English language and cultural
 2747 environment have a facility at their disposal that is not available to writers of other applications.
 2748 If any change is made, it should be to add similar facilities for other language and cultural
 2749 environments, rather than to remove the existing facility.

2750 Where there is no alternative to using the Latin alphabet, the writing of internationalised
 2751 applications, and of applications that assume language and cultural environments other than
 2752 English ones, is inhibited. In such cases it is desirable to change the specification, either to
 2753 require a more general type of string (for example, a graphic string) in place of the IA5 or
 2754 printable string, or to allow a more general type of string as an alternative. However, where the
 2755 use of the Latin alphabet string derives from an International Standard, such a change should be
 2756 made in consultation with the relevant international standards body.

2757 **18.1.2 Null-Terminated Strings**

2758 The use of null-terminated strings gives problems arising in connection with use of encoding
 2759 schemes such as ISO 10646 and UNICODE that allow embedded nulls in character encodings.

2760 This issue affects

- 2761 • the XMP, in which error message strings are null-terminated
- 2762 • the UMA data pool, in which character string message fields are null-terminated
- 2763 • the UMA logical message protocol, in which text fields are null-terminated
- 2764 • the UMA measurement layer interface, in which a number of function arguments are null-
 2765 terminated strings
- 2766 • the UMA data capture interface, in which metric data objects can be null-terminated strings
- 2767 • the XBSA, which allows variable length strings (presumably null-terminated) to be passed
 2768 across the interface
- 2769 • the Management Services for an OMG Environment, if the OMG IDL string data type is
 2770 mapped onto a null-terminated string datatype in a programming language.

2771 **18.1.3 Descriptive Text**

2772 Descriptive text assumes a particular natural language and locale. It presents problems for
 2773 system users or managers who do not understand the natural language or who do not use the
 2774 locale. For example, error messages in Japanese would be incomprehensible to most English or
 2775 American users, and could in any case probably not be displayed on their terminals.

2776 This affects

- 2777 • the error messages generated by XMP
- 2778 • textual descriptions in UMA UDU control segments
- 2779 • exception descriptions in the Management Services for an OMG Environment specification.

2780 All error messages and other textual descriptions generated by an implementation should be
2781 displayed to the user in the language of the current locale. This means that, if they have to be
2782 transmitted between different system components before being displayed to the user, they
2783 should be held internally in some coded form from which appropriate text for various locales
2784 can be generated. The message cataloguing mechanism described in XPG4 provides a means of
2785 achieving this.

2786 **18.1.4 String Comparisons**

2787 Comparisons between strings should take account of the locale (or locales) in which those
2788 strings were defined. This affects:

- 2789 • the **Filter-Item** class in XMP
- 2790 • character string names in XBSA
- 2791 • string comparisons in the Management Services for an OMG Environment specification.

2792 Ideally, comparisons should be based on information, such as collation order, that is contained
2793 in the current locale. In practice, there are a number of questions as to how this information
2794 should be interpreted, and the necessary information has not all been defined (for example,
2795 directionality information for complex text languages is still being studied).

2796 Also, some of these definitions (those in XMP, for example) are based on definitions in
2797 International Standards, and X/Open should not adopt a solution unless it is also adopted for
2798 the International Standards from which the definitions are derived.

2799 **18.1.5 Input and Output Character Sets**

2800 The input and output character sets used by language translators may restrict the locales in
2801 which the programs that they help to produce can be used. This affects the GDMO-XOM
2802 translation specification.

2803 There has been some consideration given by workers in the field of Internationalisation to
2804 characterless programming languages, in which the lexical tokens are standardised but their
2805 representations in specific character sets are not. At present, however, programming language
2806 specifications typically require particular characters to be present in the input character set.
2807 They also specify particular keywords, which may convey semantic meaning in particular
2808 language and cultural environments (usually English ones). For example, the C programming
2809 language requires the input character set to include the basic Latin alphabet, and uses the
2810 English word "if" as a keyword.

2811 For the purposes of this technical study, it is assumed that it is reasonable to require the
2812 programmer to use a particular character set and particular keywords. (After all, learning a
2813 programming language is in many ways like learning a natural language, which has its own
2814 alphabet and vocabulary.) However, the programmer must be able to have the facilities needed
2815 to produce an application that assumes any particular language and cultural environment or,
2816 preferably, is internationalised. This does not mean that the compiler must support all language
2817 and cultural environments; it means that the compiler standard does not preclude support for
2818 any particular environment.

2819 In the case of the **XGDMO** specification, the position is more complicated, because the output of
2820 the translator is not an application program; it is input to other programs, and is used by
2821 applications programmers. The principle that applies here is that it should be possible for the
2822 programs that use the output of the translator to produce an application that assumes any
2823 particular language and cultural environment or, preferably, is internationalised. This is
2824 possible with the specification in its current form. However, the recommendations in Section
2825 18.2.5 on page 124 of this technical study are made with the aim of ensuring that implementors
2826 make clear their positions with regard to support of character sets.

2827 **18.1.6 Use of specific Date/Time Formats**

2828 This affects the **XBSA** specification.

2829 Dates and times should be displayed to the user in the format prescribed by that user's current
2830 locale. They should be stored internally in a form from which any locale-dependent
2831 representation can be generated. However, as it is possible to generate locale dependent
2832 representations from any fixed format representation, the **XBSA** specification does not preclude
2833 (in this respect) the writing of internationalised applications.

2834 18.2 Recommendations

2835 18.2.1 Character String Identifiers

2836 • In XMP, class **Entity-Name** should be redefined to allow more general strings as
2837 representations of entity names.

2838 • Either:

2839 1. the character strings in the UMA Data Pool, the UMA logical message protocol, the
2840 UMA measurement layer interface, the UMA data capture interface, the
2841 Backup/Restore interface and the Management Services for an OMG Environment
2842 should be tagged with an indication of the locale in which they were created

2843 or

2844 2. users working in a multi-locale environment should be advised to use only the
2845 characters of the POSIX portable filename character set.

2846 These alternatives have been presented to the X/Open Systems Management working group.
2847 This group did not resolve to undertake the work required for alternative 1. Accordingly,
2848 alternative 2, which can easily be implemented, should be adopted.

2849 18.2.2 Null-Terminated Strings

2850 • The description of **Error-Message** in XMP should warn applications programmers not to use
2851 the null terminator unless they are sure that the character set encoding does not allow
2852 embedded nulls.

2853 • Either:

2854 1. the use of encodings that can include null bytes should be forbidden for character string
2855 message fields in the UMA data pool and the UMA logical message protocol

2856 or

2857 2. applications should be advised to rely on the length fields that are provided, rather
2858 than the null terminator, when determining the lengths of such fields.

2859 • In the UMA measurement layer interface, the UMA data capture interface and the XBSA,
2860 either:

2861 1. the use of character encodings that can include null bytes should be prohibited

2862 or

2863 2. the interface should be modified to use arrays of type **wchar_t** in place of, or as an
2864 alternative to, null-terminated arrays of type **char**.

2865 • The possibility that the OMG IDL string data type could be mapped onto a null-terminated
2866 string datatype in a programming language, and the internationalisation implications of this,
2867 should be discussed with the OMG. The impact on the Management Services for an OMG
2868 Environment specification should then be assessed.

2869 18.2.3 Descriptive Text

- 2870 • The description of **Error-Message** in XMP should state that the string returned is dependent
2871 on the current locale.
- 2872 • The use of textual descriptions in UMA UDU control segments and in OMG Environment
2873 Management Services exception descriptions should be deprecated for internationalised
2874 applications; they should be encouraged to use message catalogs.

2875 18.2.4 String Comparisons

- 2876 • No changes should be made to the specifications at this time. X/Open should work with
2877 international standards bodies to resolve the issue.

2878 18.2.5 Input and Output Character Sets

- 2879 • The **XGDMO** specification should require implementations to document the character sets
2880 that can be used for input and that will appear in the output.

Change Requests for Internationalisation

2881

2882

2883

This chapter contains formal change requests for the X/Open systems management specifications.

2884

2885

Note: Some of the change requests contained in this chapter are against pre-publication draft versions of the specifications concerned.

2886 **19.1 The XMP Specification**

2887	Document:	The XMP specification
2888		X/Open CAE Specification, P306 (March 1994).
2889	Change number:	DL-1
2890	Title:	Error Message Text Strings
2891	Qualifier:	Minor Technical
2892	Rationale:	The Error-Message function returns a text string that describes an error. Its description refers to the “X/Open Native Language System (NLS)”, but does not specify how use of the NLS affects the string returned, or constrain the character set that may be used. The length of the string is given explicitly, and the string is null-terminated. Presumably, this means that it is terminated by a single null element of type char; this means that encodings such as ISO 10646 or UNICODE that have character representations containing embedded nulls would cause applications that use the null terminator to behave incorrectly.
2900		
2901	Change:	On the <i>Error-Message()</i> man page:
2902		i. In the description of the <i>Length</i> argument, delete “This is necessary ... Native Language System).”
2903		
2904		ii. In the description of the <i>Error-text</i> result, add the following new paragraph between the two existing paragraphs:
2905		
2906		“The language and character set encoding of the message text depend on the currently established locale.”
2907		
2908		iii. In the description of the <i>Length-return</i> result, add the following new paragraph:
2909		
2910		“Internationalised applications, and other applications that use character set encodings (such as that defined by ISO 10646) that allow embedded nulls, should use the length_return value to determine the length of the message string rather than relying on the null terminator.”
2911		
2912		
2913		

2914 Document: The **XMP** specification
2915 X/Open CAE Specification, P306 (March 1994).

2916 Change number: DL-2

2917 Title: Entity name syntax

2918 Qualifier: Minor Technical

2919 Rationale: The specification of class **Entity-Name** defines attribute **entity** as a printable
2920 string. This means that management application names and system names
2921 are essentially restricted to using ASCII characters.

2922 Change: Change the Value Syntax of attribute entity of class Entity-Name from
2923 String(Printable) to String(any).

2924 **19.2 The XGDMO Specification**

2925	Document:	The XGDMO specification
2926		X/Open Preliminary Specification, P319 (March 1994).
2927	Change number:	DL-3
2928	Title:	Character Sets
2929	Qualifier:	Minor Technical
2930	Rationale:	The XGDMO specification in effect defines a language translator, and
2931		similar issues arise as for a programming language. In particular, there is the
2932		question of what character sets can be used in the input to the translator, and
2933		of what character sets can appear in its output. This question is not
2934		addressed in the XGDMO specification.
2935		It is implicitly assumed that the input character set includes the distinct
2936		lowercase and uppercase versions of the basic Latin alphabet, the decimal
2937		digits, and spacing and punctuation characters. Generated output includes
2938		specific characters (as in the "OMP_O-" prefix). OM classes and attributes are
2939		identified and alphabetised in ascending order.
2940		Lack of support for characters outside the basic Latin alphabet would mean
2941		that the natural names could not be used for managed objects and attributes
2942		that have been defined and named in a language and cultural environment
2943		other than an English one. This would affect the applications programmer,
2944		but should not affect the user of the applications programs.
2945	Change:	Add a new Section 3.6 entitled "Output Character Sets" (and renumber the
2946		existing Section 3.6 as 3.7). The new section should contain the following
2947		text:
2948		“Implementations may generate outputs using various character set
2949		encodings. Each encoding used must satisfy the conditions defined in
2950		ISO/IEC 9899 (the ISO C Standard) for source and execution character sets.
2951		Each implementation must document the character set encodings that it
2952		uses.”

2953 **19.3 The UMA Specifications**

2954 Documents: The **UMA DPD** specification
 2955 X/Open Preliminary Specification, P435 (Pre-publication Draft).

2956 The **UMA DCI** specification
 2957 X/Open Preliminary Specification, P434 (Pre-publication Draft).

2958 The **UMA MLI** specification
 2959 X/Open Preliminary Specification, P426 (Pre-publication Draft).

2960 Change number: DL-4

2961 Title: Use of Text Strings in Messages

2962 Qualifier: Minor Technical

2963 Rationale: Character strings are used in fields of data pool messages, in the Logical
 2964 Message Protocol, in Data Capture Interface data types and in Data Capture
 2965 Interface labels. It is not clear what will happen if different programs assume
 2966 different character set encodings when processing these strings.

2967 It might be possible to solve this problem in a way that allows for fully
 2968 internationalised applications using arbitrary character set encodings. For
 2969 example, all text strings in messages might be replaced by fields encoded
 2970 using an extension of the heirarchical scheme used in the Data Capture
 2971 metric name space. This would require a major revision of the whole UMA.
 2972 The following change is proposed on the assumption that such a major
 2973 revision will not be undertaken.

2974 Change:

2975 i. In each of the following sections:

2976 a. Section 2.4.3 (UMA Type Definitions) of the **UMA DPD**
 2977 specification, after the definition of *UMATextDescr*,

2978 b. Chapter 6 (UMA Message and Header Formats) of the **UMA MLI**
 2979 specification, before the start of section 6.1,

2980 c. Section 3.3.1.1 (DCILabel) of the **UMA DCI** specification, after the
 2981 definition of the *DCILabel* structure (and see also below for other
 2982 changes to this section), and,

2983 d. Section 3.3.4 (Data Types) of the **UMA DCI** specification, after the
 2984 definition of *DCITextDescr*,

2985 insert the following new paragraph:

2986 “Note that all programs that use this data must assume the same
 2987 character set and encoding scheme. Where different character sets are
 2988 in use (for example, in distributed multinational systems), it is
 2989 recommended that text strings use only the characters of the POSIX
 2990 portable file name character set. It is recognised that this pragmatic
 2991 recommendation places some limitations on the degree to which
 2992 applications can be internationalised.”

2993 ii. In Section 1.3.1 (Goals) of the **UMA DCI** specification, delete the
 2994 "internationalisation" list item.

- 2995
2996
- 2997
2998
2999
- 3000
3001
3002
3003
3004
- 3005
3006
- 3007
- 3008
3009
- 3010
3011
- 3012
3013
3014
- iii. Delete Section 1.3.10 (Internationalisation) of the **UMA DCI** specification.
 - iv. In Section 3.3.1.1 (*DCILabel*) of the **UMA DCI** specification, change “The label attributes structure ... The *DCILabel* structure is:” to the following text:
“The label attributes structure contains a variable length text field (an array of type *char*, null-terminated) and a field that gives the size of the text field (the number of elements in the array, including the null terminator). The field must be padded out to a four byte boundary. The *DCILabel* structure is:”
 - v. In the definition of *DCILabel* in Section 3.3.1.1 (*DCILabel*) of the **UMA DCI** specification:
 - a. Delete the line defining field “ascii”
 - b. On the next line (defining field “i18n”), change “i18n” to “desc” and delete “ for I18N”
 - c. On the following line (defining field “data”), delete “ for ascii and i18n”
 - vi. Delete the paragraph following the definition of *DCILabel* in Section 3.3.1.1 (*DCILabel*) of the **UMA DCI** specification (“The internationalised label ... four byte boundary.”)

3015	Documents:	The UMA DPD specification
3016		X/Open Preliminary Specification, P435 (Pre-publication Draft).
3017		The UMA DCI specification
3018		X/Open Preliminary Specification, P434 (Pre-publication Draft).
3019	The UMA MLI specification	
3020		X/Open Preliminary Specification, P426 (Pre-publication Draft).
3021	Change number:	DL-5
3022	Title:	Null-Terminated Message Strings
3023	Qualifier:	Minor Technical
3024	Rationale:	When null-terminated strings are used, the use of encodings such as ISO
3025		10646 or UNICODE that include nulls in character encodings is problematic.
3026		The specification should either preclude the use of such encodings or advise
3027		applications to rely on the length field that is provided, rather than the null
3028		terminator, when determining the lengths of such fields.
3029	Change:	Two (mutually exclusive) alternatives are proposed. They affect the
3030		following text:
3031		a. Section 2.4.3 (UMA Type Definitions) of the UMA DPD specification,
3032		after the definition of <i>UMATextDescr</i>
3033		b. Chapter 6 (UMA Message and Header Formats) of the UMA MLI
3034		specification, before the start of Section 6.1
3035		c. Section 3.3.1.1 (DCILabel) of the UMA DCI specification, after the
3036		definition of the <i>DCILabel</i> structure (and see also below for other
3037		changes to this section)
3038		d. Section 3.3.4 (Data Types) of the UMA DCI specification, after the
3039		definition of <i>DCITextDescr</i> .
3040		The alternatives are:
3041		i. When making the change requested by CR DL-4, use the following text
3042		in place of that given in CR DL-4:
3043		“Note that all programs that use this data must assume the same
3044		character set and encoding scheme. This scheme must not allow
3045		embedded nulls in character encodings. Where different character sets
3046		are in use (for example, in distributed multinational systems), it is
3047		recommended that text strings use only the characters of the POSIX
3048		portable file name character set. It is recognised that these pragmatic
3049		recommendations place some limitations on the degree to which
3050		applications can be internationalised.”
3051		ii. Insert the following text, as a separate paragraph, before the text
3052		requested by CR DL-4:
3053		“Some character set encodings allow embedded nulls. Unless an
3054		application can assume that it is not dealing with such an encoding, it
3055		should not rely on the null terminator to determine the length of the
3056		string.”

3057 Document: The **UMA MLI** specification
3058 X/Open Preliminary Specification, P426 (Pre-publication Draft).

3059 Change number: DL-6

3060 Title: MLI Function Arguments

3061 Qualifier: Minor Technical

3062 Rationale: A number of arguments of the functions defined in the **UMA MLI**
3063 specification are character strings. It is not clear what character set
3064 encodings can be used for these strings.

3065 Change: In Section 5.2 (MLI Call Parameters), insert the following paragraph before
3066 the description of the first parameter (*attrpairs*).

3067 “For string parameters, the character set and encoding used will be that of
3068 the currently established locale. Note that all programs that use this data
3069 must assume the same character set and encoding scheme. Where different
3070 character sets are in use (for example, in distributed multinational systems),
3071 it is recommended that text strings use only the characters of the POSIX
3072 portable file name character set. It is recognised that this pragmatic
3073 recommendation places some limitations on the degree to which applications
3074 can be internationalised.”

3075	Document:	The UMA MLI specification
3076		X/Open Preliminary Specification, P426 (Pre-publication Draft).
3077	Change number:	DL-7
3078	Title:	Null-Terminated Function Arguments
3079	Qualifier:	Minor Technical
3080	Rationale:	When null-terminated strings are used, the use of encodings such as ISO
3081		10646 or UNICODE that include nulls in character encodings is problematic.
3082	Change:	Three mutually exclusive possible changes are proposed:
3083		i. When making the change to Section 5.2 (MLI Call Parameters)
3084		requested by CR DL-6, use the following text in place of that given in
3085		CR DL-6:
3086		“For string parameters, the character set and encoding used will be that
3087		of the currently established locale. The encoding scheme must not
3088		allow embedded nulls in character encodings. Note that all programs
3089		that use this data must assume the same character set and encoding
3090		scheme. Where different character sets are in use (for example, in
3091		distributed multinational systems), it is recommended that text strings
3092		use only the characters of the POSIX portable file name character set. It
3093		is recognised that these pragmatic recommendations place some
3094		limitations on the degree to which applications can be
3095		internationalised.”
3096		ii. Change all function arguments that are of type char* to be of type
3097		wchar_t* .
3098		iii. For each function that has arguments that are of type char* , keep that
3099		function unchanged, but introduce an additional function that
3100		performs the same tasks but with the char* arguments replaced by
3101		arguments of type wchar_t* .

3102 Document: The **UMA MLI** specification
3103 X/Open Preliminary Specification, P426 (Pre-publication Draft).

3104 Change number: DL-8

3105 Title: UDU Error Descriptions

3106 Qualifier: Minor Technical

3107 Rationale: The bodies of UDU control segments can contain textual descriptions of
3108 problems encountered. These are not usable by internationalised
3109 applications.

3110 Change: In Section 6.2 (UDU Control Segments) replace the text:
3111 “In addition, ... user’s terminal”
3112 by
3113 “The application should use the catalogue mechanism to generate a textual
3114 description of the problem in the language of the currently established
3115 locale”.

3116 **19.4 The XBSA Specification**

3117	Document:	The XBSA specification
3118		X/Open Preliminary Specification, P424 (Pre-publication Draft).
3119	Change number:	DL-9
3120	Title:	Use of Character Strings
3121	Qualifier:	Minor Technical
3122	Rationale:	Names of users, files and other entities (such as domains and policy sets) are
3123		passed across the API in character strings. Different users and applications
3124		will create these names in different locales. Difficulties will arise if the locale
3125		assumed by the implementation to process a name is not the same as the
3126		locale in which the name was created. This could happen if, for example, a
3127		system administration facility running in one locale is used to archive or
3128		restore files that have been created (and named) in other locales.
3129	Change:	Add a new Section 2.7 entitled “Internationalisation” (and renumber the
3130		remaining subsections of Section 2). The new section should contain the
3131		following text:
3132		“Names of users, files and other entities (such as domains and policy sets)
3133		are passed across the API in character strings. Different users and
3134		applications will create these names in different locales. Difficulties will
3135		arise if the locale assumed by the implementation to process a name is not
3136		the same as the locale in which the name was created. This could happen if,
3137		for example, a system administration facility running in one locale is used to
3138		backup, archive or restore files that have been created (and named) in other
3139		locales.”
3140		“Implementations of the XBSA will process character strings in a locale-
3141		dependent manner. Each function will assume the locale that is established
3142		at the time that it is invoked. Applications should ensure that the
3143		appropriate locale is established when an XBSA function is called.”

3144	Document:	The XBSA specification
3145		X/Open Preliminary Specification, P424 (Pre-publication Draft).
3146	Change number:	DL-10
3147	Title:	Null-terminated strings
3148	Qualifier:	Minor Technical
3149	Rationale:	Although the XBSA specification does not say so, the variable length
3150		character strings that are passed across the API are presumably null-
3151		terminated. This will give problems if the characters are encoded in
3152		UNICODE or ISO 10646.
3153	Change:	Three mutually exclusive possible changes are proposed:
3154		i. When adding the new Section 2.7 requested by Change Request (CR)
3155		DL-9, include in it the following text in addition to that given in CR
3156		DL-9:
3157		“The encoding scheme of the established locale must not allow
3158		embedded nulls in character encodings.”
3159		ii. Change “char” to “wchar_t” in the definitions of the following types:
3160		Administrator
3161		AdminName
3162		AppUserName
3163		BSAUserName
3164		CGName
3165		CopyGPDest
3166		CopyGPName
3167		Description
3168		DomainName
3169		EventInfo
3170		FilterRuleSet
3171		LGName
3172		MethodName
3173		ObjInfo
3174		ObjectName
3175		PolicySetName
3176		ResourceType
3177		iii. For each function that has arguments that include arrays of characters,
3178		keep that function unchanged, but introduce an additional function
3179		that performs the same tasks but with the arguments replaced by
3180		arguments that have arrays of type wchar_t in place of the arrays of
3181		type char .

3182 **19.5 The XSMS Specification**

3183 Document: The XSMS specification
 3184 X/Open Preliminary Specification, P421 (Pre-publication Draft).

3185 Change number: DL-11

3186 Title: Use of Text Strings

3187 Qualifier: Minor Technical

3188 Rationale: Character strings are used in the following:

3189 — Names of objects, object types, filters, actions and other entities

3190 — Arguments to commands in the command line interface

3191 — Exception data types.

3192 Ideally, there should be provision for internationalised applications to use
 3193 Management Services in an OMG Environment. The possibilities of different
 3194 users requiring different character sets and encodings, and the possibilities
 3195 that these encodings could allow embedded nulls, should be taken into
 3196 account in the architecture of Management Services in an OMG
 3197 Environment. At present, the architecture does not take these possibilities
 3198 into account. There are complex issues involved, and the matter requires
 3199 further study.

3200 Change:

3201 i. Change the title of Chapter 4 to “Components and Issues Not
 3202 Addressed”.

3203 ii. Add a new Section 4.3 entitled “Internationalisation”. The new section
 3204 should contain the following text:

3205 “There should be provision for internationalised applications to use
 3206 Management Services in an OMG Environment. Character strings are
 3207 used in:

3208 — Names of objects, object types, filters, actions and other entities

3209 — Arguments to commands in the command line interface

3210 — Exception data types.”

3211 “The possibilities of different users requiring different character sets
 3212 and encodings, and the possibility that these encodings could allow
 3213 embedded nulls, should be taken into account. At present, the
 3214 architecture does not take these possibilities into account. There are
 3215 complex issues involved, and the matter requires further study.”

3216 “Meanwhile:

3217 — Implementations should document any restrictions on character
 3218 sets and encodings that they impose.

3219 — Where different character sets are in use (for example, in distributed
 3220 multinational systems), it is recommended that text strings use only
 3221 the characters of the POSIX portable file name character set.

3222 — Applications that may be used internationally should not use the
 3223 textual descriptions in exception descriptions but should establish

3224

and use locale-dependent message catalogues instead.’

3225

“It is recognised that this pragmatic recommendation places limitations on the degree to which applications can be internationalised.”

3226

3227

3228 / *X/Open Technical Study*

3229 **Part 6**

3230 **Glossary and Index**

3231 *X/Open Company Ltd.*

Glossary

3232

- 3233 **ANSI**
3234 American National Standards Institute
- 3235 **ANSI C**
3236 American National Standards Institute specification of the C programming language
- 3237 **API**
3238 In X/Open, an Application Programming Interface. This is a set of services (such as
3239 functions in a given programming language) by which the application program
3240 communicates with other software components.
- 3241 **ASCII**
3242 American Standard Code for Interchange of Information. It is a 7-bit code with no parity
3243 recommendation, providing 128 different bit patterns for character representation. The
3244 internationally agreed version is called ISO-7 and is specified in ISO/IEC 646.
- 3245 **BMP**
3246 Basic Multilingual Plane. ISO IS 10646 is intended to be able to cover the character sets of all
3247 languages which may be used in conjunction with computer systems. It defines a four-octet
3248 representation for each character. The characters whose representations have zero as their
3249 two most significant octets form what is known as the Basic Multilingual Plane (this
3250 includes most alphabetic character sets).
- 3251 **byte**
3252 A binary expression forming a basic character combination that usually, but not always,
3253 comprises 8 bits.
- 3254 **CAE**
3255 X/Open's Common Applications Environment.
- 3256 **CCITT**
3257 (Consultative Committee of International Telegraph and Telephone) An international
3258 committee whose membership is largely composed of government postal, telephone and
3259 telegraph agencies (PTTs). This body is now a division of the ITU, and is now called the
3260 ITU-T.
- 3261 **Change Request (CR)**
3262 In X/Open, a formal presentation of a request to change a document. It has a prescribed set
3263 of headings, which identify the document, the originator, the subject, the qualifier
3264 (critical/major/minor and technical/editorial), the rationale for the change proposal, and
3265 the proposed detailed change(s).
- 3266 **codeset**
3267 The bit patterns that constitute the encodings of a character set.
- 3268 **CR**
3269 See **Change Request**.
- 3270 **FSS-UTF**
3271 UCS Transformation Format. an algorithm which, when applied to an IS 10646 encoding,
3272 yields a 1, 2, 3 or 5 octet value which is guaranteed not to contain the ISO 646 encodings of
3273 any control character, or of the SPACE or DEL characters.

- 3274 **IEC**
3275 International Electrotechnical Commission.
- 3276 **IEEE**
3277 In U.S.A., the Institute of Electrical and Electronic Engineers.
- 3278 **interoperability**
3279 The ability of software and hardware on multiple machines and from multiple vendors to
3280 work together effectively.
- 3281 **internationalization**
3282 The provision within a computer program of the capability to make itself adaptable to the
3283 requirements of different native languages, cultural environments and coded character sets.
- 3284 **ISO**
3285 International Organisation for Standardisation. A standards organisation with the
3286 membership composed of the standards organisations from each participating country. ISO
3287 working groups generate the OSI Protocol Suite standards.
- 3288 **ISO C**
3289 The ISO definition of the C programming language, technically the same as ANSI C.
- 3290 **ITU**
3291 International Telecommunications Union. See also **CCITT**.
- 3292 **I18n**
3293 Abbreviation for Internationalization, there being 18 letters between the first and last letters
3294 in this long word.
- 3295 **locale**
3296 The definition of the subset of a user's environment that depends on language and cultural
3297 conventions.
- 3298 **octet**
3299 8 contiguous bits. The term is used instead of byte to prevent confusion with machines that
3300 use non-8-bit bytes.
- 3301 **OSI**
3302 Open Systems Interconnection. A set of ISO standards for the interconnection of
3303 cooperative (open) computer systems, using the ISO 7-layer reference model.
- 3304 **portability**
3305 Machine-independent — applied to software which can be readily ported to different
3306 machines.
- 3307 **POSIX**
3308 A set of IEEE and ISO standards for a portable operating system, based on UNIX.
- 3309 **presentation**
3310 OSI Presentation Layer — the 6th layer in the 7-layer OSI Reference Model. It preserves the
3311 meaning of the data transferred between Application Entities, and also provides access to
3312 the services of the Session Layer.

- 3313 **protocol**
3314 A specification for an agreed procedure to enable exchange of information between
3315 cooperating entities, via interfaces which provide the necessary functionality to cover
3316 format of messages, data checks, flow control, and error handling. A set of protocols
3317 governing the exchange of information between remote systems, and set of interfaces
3318 covering the exchange between adjacent protocol levels, are collectively referred to as a
3319 protocol hierarchy or protocol stack.
- 3320 **teletype**
3321 An electrical typewriter machine equipped with a signal interface through which it provides
3322 hard copy output and keyboard input for a computer.
- 3323 **UCS**
3324 Universal Multiple-Octet Coded Character Set — defined in ISO 10646.
- 3325 **User Datagram Protocol**
3326 The connectionless transport layer protocol of the Internet Protocol Suite.
- 3327 **UTF-8**
3328 See **FSS-UTF**.
- 3329 **XPG**
3330 X/Open Portability Guide
- 3331 **XSI**
3332 X/Open System Interface (to an operating system).

Index

1

2	(PC)NFS specification	19, 39	XA specification.....	95
3	Change Request (CR)	57	XA+ specification	98
4	 character	8	XAP specification	48
5	<ETX> character	8	XBSA specification	135
6	<SPACE> character	8	XDS specification.....	55
7	<tx.h> header	86, 92	XFTAM specification	53
8	<xa.h> header.....	87, 89, 92	XGDMO specification	128
9	/ character	8	XMP specification	126
10	ACSE/Presentation APIs.....	26	XNFS specification	57
11	action entity	116	XOM specification.....	50
12	alphabetic classification	5	XSMS specification.....	137
13	ANSI.....	141	XTI specification	46
14	ANSI C.....	9, 141	char	9
15	API.....	141	character	9
16	application		8
17	communicating with other application	13	<ETX>	8
18	archive.....	115	<SPACE>.....	8
19	argument.....	85, 87, 89, 91	composition	14
20	arithmetical expression	69	directionality	14
21	array descriptor.....	112	lower case.....	5
22	ASCII.....	6, 9, 141	null.....	9
23	attribute.....	107, 115	shaping	14
24	attribute character sets	34	upper case	5
25	backup.....	115	/.....	8
26	Backup Services API (XBSA).....	115	character case conversion	5
27	Basic Multilingual Plane.....	7	character classification	5
28	BMP	141	character encoding	6, 65
29	BSFT specification	19, 30	character set	5-6, 21, 65
30	Change Request (CR)	52	input.....	121
31	business practice	5	output	121
32	byte	141	character set register	6
33	C.....	117, 121	character string conversion	
34	C programming language.....	5, 9	CLI specification	73
35	char	9	character string passing	
36	C++	117	CLI specification	72
37	CAE	141	character string type	27
38	case convention	5	characteristic	85, 87, 89, 91
39	CCITT	6, 141	class	107
40	Change Request (CR) ...	19, 45, 83, 93, 105, 125, 141	disk device	112
41	(PC)NFS specification	57	Entity-Name	123
42	BSFT specification	52	class name	
43	IPC mechanisms for SMB specification	57	response time	112
44	SMB protocols specification.....	57	classification rules	10
45	TX specification	94	CLI specification.....	61, 72
46	UMA specifications.....	129	character string conversion	73
47	X.400 API specification.....	54	character string passing	72

48	internationalisation implications	72	ISO 10646 and C	67
49	COBOL	85	multiple character set	64
50	codeset	6, 141	non-direct invocation	66
51	collation rules	5, 10	numeric literal	68
52	command line interface	119	reserved word	68
53	Common Usage C	85, 87, 89, 92	special character	68
54	communication		standard name	64
55	applications	13	string operation	67
56	communications interface	21	text editor	66
57	comparison of strings	117, 121	data management specification	61, 71
58	complex text languages	14	CLI specification	61
59	composition of characters	14	RDA specification	61
60	control character	6-7	SQL	61, 63
61	CPI-C specification	83	data pool message field	112
62	CR	19, 45, 83, 93, 105, 125, 141	data services layer	112
63	(PC)NFS specification	57	data type	
64	BSFT specification	52	data	capture
65	IPC mechanisms for SMB specification	57	date	115, 122
66	SMB protocols specification	57	date format	9
67	TX specification	94	date literal	68
68	UMA specifications	129	date representation	5
69	X.400 API specification	54	decimal separator	5
70	XA specification	95	delete () character	8
71	XA+ specification	98	descriptor	
72	XAP specification	48	array	112
73	XBSA specification	135	text	112
74	XDS specification	55	destination field of Create Message	112
75	XFTAM specification	53	diagnostic information	68
76	XGDMO specification	128	diagnostic message	26
77	XMP specification	126	diagnostic text	29
78	XNFS specification	57	direct invocation	66
79	XOM specification	50	directionality	5, 69
80	XSMS specification	137	of	14
81	XTI specification	46	disk device class	112
82	Create message		disk partition data subclass	112
83	destination field	112	distributed internationalisation	12
84	source field	112	domain name	115
85	cultural convention	5, 9	DTP specification	83, 85
86	currency symbol	5	CPI-C specification	83
87	data capture interface	112-113	TX specification	83
88	label	113	TxRPC specification	83
89	data type	113	XA specification	83
90	data management		XA+ specification	83
91	arithmetical expression	69	XATMI specification	83
92	character encoding	65	encoding	6, 21, 40
93	character set	65	UCS-2 form	7, 9
94	date literal	68	UCS-2 Level 3	7
95	diagnostic information	68	UCS-4 form	7, 9
96	direct invocation	66	End of Text (<ETX>) character	8
97	directionality	69	entity	107
98	host language processor	66	action	116

Index

99	filter	116	interoperability	21, 142
100	Entity-Name class	123	interworking	22
101	entry level SQL	63	interworking specification	19, 23
102	error message	26, 107	(PC)NFS	19
103	Error-Message in XMP	123-124	BSFT	19
104	exception condition in IDL	117	IPC mechanisms for SMB	19
105	file	115	SMB	19
106	file content	29	X.400 API	19
107	file type	30	XAP	19
108	filename	115	XAP-ROSE	19
109	filter entity	116	XAP-TP	19
110	filter operation	117	XDS	19
111	Filter-Item class in XMP	121	XFTAM	19
112	four-octet representation	7	XMPTN	19
113	framework	15	XMS	19
114	FSS-UTF	141	XNFS	19
115	FSS-UTF algorithm	8	XOM	19
116	full SQL	63	XTI	19
117	function Name	85, 87, 89, 91	introduction	3
118	GDMO to XOM Translation Algorithm	110	IPC mechanisms for SMB	19, 42
119	graphic character	6	IPC mechanisms for SMB specification	
120	composite symbol	7	Change Request (CR)	57
121	graphic string	120	ISO	6, 142
122	host language processor	66	ISO 10646	15, 43
123	I18n	142	ISO 10646 and C	67
124	I18n change request	19, 83, 105	ISO C	9, 85, 87, 89, 92, 142
125	I18n considerations	5	null character	9
126	IA5	107, 120	ITU	142
127	identifier string	29	JIG	8
128	IDL	117, 119	Joint Internationalisation Group (JIG)	8
129	exception condition	117	language	5-6, 9, 22, 37
130	IDL in OMG	116, 123	C programming language	9
131	IEC	142	language translator	121
132	IEEE	142	Latin alphabet	120-121
133	information processing	5	string	107
134	information-creator name	115	locale	9, 11-12, 43, 115-116, 142
135	input character set	121	conveying information	13
136	intermediate SQL	63	conveying information between	13
137	international alphabet 5	107	use by application	13
138	International Alphabet 5 (IA5)	120	locale definition	14
139	internationalisation		locale registration	14
140	conclusions	43, 77, 91, 119	logical message protocol	112
141	recommendations	43, 77, 91, 123	lower case character	5
142	internationalisation considerations	5	managed objects	110
143	internationalisation framework	15	templates	110
144	internationalisation implications		Management Protocol Profiles (XMPP)	109
145	CLI specification	72	Management Protocols API (XMP)	107
146	RDA specification	74	Management Services for OMG	116, 119, 123
147	SQL specification	71	MAP	114
148	internationalization	142	measurement layer interface	113
149	Internet protocol	6	string argument	113

- 150 message120
- 151 message catalogue in XPG4121
- 152 migration12
- 153 MPTN25
- 154 multi-byte character9
- 155 multi-locale43
- 156 multi-locale support14
- 157 multiple character set64
- 158 name31, 116
- 159 native language system (NLS)107
- 160 natural language text31
- 161 NetBIOS24
- 162 NLS107
- 163 non-direct invocation66
- 164 null character9
- 165 null terminator123
- 166 null-terminated string117, 120
- 167 number representation5, 14
- 168 numeric classification5
- 169 numeric literal68
- 170 object in OMG116
- 171 object management116
- 172 object-oriented programming116
- 173 octet6, 142
- 174 four-octet representation7
- 175 OMG
 - 176 IDL116, 123
 - 177 Management Services119, 123
- 178 OMG object116
- 179 OMG object management116
- 180 open systems environment5
- 181 OSI142
- 182 OSI protocol6
- 183 output character set121
- 184 PDU31
- 185 policy set name115
- 186 portability21, 142
- 187 Portable Filename Character Set11, 22
- 188 portable filename character set
 - 189 POSIX123
- 190 POSIX5, 142
 - 191 Portable Filename Character Set11
 - 192 portable filename character set123
- 193 POSIX locale mechanism11
- 194 presentation142
- 195 process environment11
- 196 protocol37, 40, 143
 - 197 Internet6
 - 198 OSI6
- 199 punctuation5
- 200 RDA specification61, 74
 - internationalisation implications74
 - visible string74
- remote terminal monitor measures subclass112
- reserved word68
- response time class112
- return Code85, 87, 89, 91
- service provider26
- shaping of characters14
- shift rules5
- slash (/) character8
- SMB
 - Protocols40
 - SMB data40
 - SMB host names40
 - SMB management transaction41
 - SMB protocols specification
 - Change Request (CR)57
 - SMB specification19, 40, 42
 - source field of Create message112
 - space (<SPACE>) character8
 - special character68
 - specification105
- SQL63
 - entry level63
 - full63
 - intermediate63
- SQL specification61, 71
 - internationalisation implications71
- standard name64
- string9, 113, 115-117, 119
 - graphic120
 - identifier119
 - Latin alphabet107
 - null terminator123
 - null-terminated117, 120
- string argument113
- string comparison107, 117, 121
- string operation67
- subclass
 - disk partition data112
- subclass name
 - remote terminal monitor measures112
- systems management
 - reference model116
 - specification105, 107
- Teletex7
- teletype6, 143
- templates
 - managed objects110
- testing15
- text113, 120

Index

201	text descriptor	112	BSFT specification	19, 30, 52
202	text editor	66	Change Request (CR)	19, 45, 83, 93, 105, 125
203	TFA	37	CLI specification	61, 72
204	time	9, 115, 122	CPI-C specification	83
205	time of day representation	5	data management specification	61, 71
206	transparent file access	37	DTP specification	83, 85
207	TX specification	83, 85	interworking specification	19, 23
208	Change Request (CR)	94	IPC mechanisms for SMB	19
209	TxRPC specification	83	IPC mechanisms for SMB specification	57
210	type mapping	116	Portability Guide (XPG)	12
211	t_error	23	RDA specification	61, 74
212	t_strerror	24	SMB protocols specification	57
213	UCS	143	SMB specification	19, 40, 42
214	UCS Transformation Format (UTF)	8	specification	105
215	UCS-2 encoding	7, 9	SQL specification	61, 71
216	UCS-2 Level 3 encoding	7	systems management specification	105, 107
217	UCS-4 encoding	7, 9	TX specification	83, 85, 94
218	UDU	113, 124	TxRPC specification	83
219	UMA		UMA specifications	111, 129
220	backup/restore interface	119, 123	X.400 API specification	19, 32, 54
221	data capture interface	119, 123	XA specification	83, 87, 95
222	data pool	119, 123	XA+ specification	83, 89, 98
223	logical message protocol	119, 123	XAP specification	19, 26, 48
224	measurement layer interface	119, 123	XAP-ROSE specification	19, 26
225	UMA Data Unit (UDU)	113, 124	XAP-TP specification	19, 26
226	UMA specifications	111	XATMI specification	83
227	Change Request (CR)	129	XBSA specification	115, 135
228	UNICODE	7, 15, 43	XDS specification	19, 34, 55
229	Universal Measurement Architecture (UMA) ..	111	XFTAM specification	19, 29, 53
230	upper case character	5	XGDMO specification	110, 128
231	User Datagram Protocol	143	XMP specification	107, 126
232	user interface	5, 21, 30	XMPP specification	109
233	user name	115	XMPTN specification	19, 25
234	UTF	8	XMS specification	19, 33
235	UTF-1 algorithm	8	XNFS specification	19, 36, 57
236	UTF-2 algorithm	8	XOM specification	19, 27, 50
237	UTF-8	143	XSMS specification	116, 137
238	variable-locale	43	XTI specification	19, 23, 46
239	Videotex	7	X/Open-Uniform	
240	visible string		Joint Internationalisation Group	8
241	RDA specification	74	XA specification	83, 87
242	wchar_t	9, 12, 43, 91	Change Request (CR)	95
243	wide character	91	XA+ specification	83, 89
244	wide characters	9	Change Request (CR)	98
245	wildcard	115	XAP specification	19, 26
246	worldwide portability interface	12	Change Request (CR)	48
247	X.400 API specification	19, 32	XAP-ROSE specification	19, 26
248	Change Request (CR)	54	XAP-TP specification	19, 26
249	X/Open		XATMI specification	83
250	(PC)NFS specification	19, 39, 57	XBSA specification	115
251	ACSE/Presentation APIs	26	Change Request (CR)	135

252	XDS specification.....	19, 34
253	Change Request (CR)	55
254	XFTAM specification	19, 29
255	Change Request (CR)	53
256	XGDMO specification	110
257	Change Request (CR)	128
258	XID structure	86-87, 89
259	XMP specification.....	107
260	Change Request (CR)	126
261	Error-Message	123-124
262	Filter-Item class.....	121
263	XMPP specification	109
264	XMPTN specification	19, 25
265	XMS specification	19, 33
266	XNFS specification	19, 36
267	Change Request (CR)	57
268	XOM specification.....	19, 27
269	Change Request (CR)	50
270	XPG	12, 143
271	XPG4	
272	message catalogue	121
273	XPG4 facilities	12
274	XSI.....	143
275	XSMS specification.....	116
276	Change Request (CR)	137
277	XTI specification	19, 23
278	Change Request (CR)	46