-- --

IEEE P1003.0 Draft 13 – September 1991

2.2.1.3 may: An indication of an optional feature.

With respect to implementations, the word *may* is to be interpreted as an optional feature that is not required in this guide, but can be provided.

2.2.1.4 normative: Of, pertaining to, or prescribing a norm or standard. D

Used in standards to indicate a portion of the text that poses requirements. D

2.2.1.5 should: With respect to implementations, an indication of an implementation recommendation, but not a requirement.

2.2.1.6 POSIX : The term ''POSIX'' has been evolving recently into a generally positive term with, unfortunately, a number of different meanings. This subclause attempts to define the word and some related terms. The intent is to insure that the term POSIX is used in a useful and predictable manner in this document.

As background, note that POSIX is sometimes used to denote the formal standard IEEE Standard 1003.1-1990, sometimes to denote that standard plus related standards and drafts emerging from IEEE P1003.x working groups, and sometimes to denote the groups themselves. In all those cases, it should be noted, POSIX is used as a noun.

This document will use the term ''POSIX''only as an adjective, and will use it only in well defined ways. This subclause serves as a preview of the usages in this book of POSIX terms. (These terms are defined, formally, or informally in subsequent clauses, and you will be referred to those clauses as appropriate.)

The original POSIX standard will be referred to by its name, ISO 9945, and not by the term POSIX.

The IEEE groups developing standards related to ISO 9945 are called, in this document, *POSIX working groups* . Examples are the IEEE working groups P1003.2, P1003.3, etc. The groups' names will be abbreviated POSIX. 2, POSIX. 3, etc.

The standards emerging out of the POSIX working groups will be referred to by their formal names (e.g., IEEE P1003.2 Draft 9) and are called either *POSIX Base Standards* or *POSIX Standardized Profiles* (POSIX SP s).

2.2.2 General Terms

For the purposes of this guide, the following definitions apply:

2.2.2.1 application: The use of capabilities (services/facilities) provided by an information system specific to the satisfaction of a set of user requirements.

NOTE: These capabilities include hardware, software, and data.

2.2.2.2 application platform: A set of resources that support the services on which an application or application software will run.

The application platform provides services at its interfaces that, as much as possible, make the specific characteristics of the platform transparent to the application.

2.2.2.3 application program interface (API): The interface between the applications software and the applications platform, across which all services are provided.

The application program interface is primarily in support of application portability, but system and application interoperability are also supported via the communications API.

2.2.2.4 application software: Software that is specific to an application and is composed of programs, data, and documentation.

2.2.2.5 Applications Environment Profile (AEP): A profile, specifying a complete and coherent subset of an OSE, in which standards, options, and parameters chosen are necessary to support a class of applications.

AEPs are intended for use in procurement, conformance testing, and designating a software engineering target.

2.2.2.6 base standard: A standard or specification that is recognized as appropriate for normative reference in a profile by the body adopting that profile.

2.2.2.7 Communications Interface: The boundary between application software and the external environment, such as other application software, external data transport facilities, and devices.

The services provided are those whose protocol state, syntax, and format all must be standardized for interoperability.

2.2.2.8 domain: A field of influence, such as office or industrial automation, transaction processing, or realtime control systems.

2.2.2.9 External Environment Interface (EEI): The interface between the application platform and the external environment across which information is exchanged.

The External Environment Interface is defined primarily in support of system and application interoperability.

The primary services present at the External Environment Interface comprise:

— Human/Computer Interaction Services

— Information Services

— Communications Services

2.2.2.10 external environment: A set of external entities to the application platform in which information is exchanged.

These devices include displays, disk drives, sensors, and effectors directly accessible within the system.

2.2.2.11 hardware: Physical equipment used in data processing as opposed to programs, procedures, rules, and associated documentation.

2.2.2.12 Human/Computer Interface: The boundary across which physical interaction between a human being and the application platform takes place.

2.2.2.13 Information Interchange Interface: The boundary across which external, persistent storage service is provided.

Only the format is required to be specified for data portability and interoperability.

2.2.2.14 interface: The shared boundary between two functional units, defined by functional characteristics and other characteristics, as appropriate.

2.2.2.15 internationalization: The process of designing and developing a product with a set of features, functions, and options intended to facilitate the adaptation of the product to satisfy a variety of cultural environments.

2.2.2.16 interoperability: The ability of two or more systems to exchange information and to mutually use the information that has been exchanged.

2.2.2.17 language-binding API: The interface between applications and appli- D
cation platforms based on language-independent binding APIs and consistent with D
the paradigms used for a specific programming language. D

2.2.2.18 language-independent service specification: A specification that D
facilitates the management and development of consistent language-binding stan- D
dards. D

2.2.2.19 locale: A description of a cultural environment.

2.2.2.20 localization: The process of utilizing the internationalization features to create a version of the product for a specific culture.

2.2.2.21 local adaptation: The process of modifying a product that has hard-coded biases of one culture to the hard-coded biases of another culture.

2.2.2.22 open specifications: Public specifications that are maintained by an open, public consensus process to accommodate new technologies over time and that are consistent with international standards.

2.2.2.23 Open System Application Program Interface: A combination of standards-based interfaces specifying a complete interface between an application program and the underlying application platform.

This is divided into the following parts:

— Human/Computer Interaction Services API

— Information Services API

— Communications Services API

— System Services API

2.2.2.24 open system: A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software:

— to be ported with minimal changes across a wide range of systems

— to interoperate with other applications on local and remote systems

— to interact with users in a style that facilitates user portability.

2.2.2.25 Open System Environment (OSE): The comprehensive set of inter- D
faces, services, and supporting formats, plus user aspects for interoperability or D
for portability of applications, data, or people, as specified by information technol- D
ogy standards and profiles. D

2.2.2.26 performance: A measure of a computer system or subsystem to perform its functions; for example, response time, throughput, number of transactions per second.

2.2.2.27 performance evaluation: The technical assessment of a system or system component to determine how effectively operating objectives have been achieved.

2.2.2.28 performance requirement: A requirement that specifies a performance characteristic that a system or system component must possess; for example, speed, accuracy, frequency.

2.2.2.29 portability: The ease with which software can be transferred from one information system to another.

2.2.2.30 POSIX Open System Environment (POSIX OSE): The Open System Environment in which the standards included are International, Regional, and National Information Technology Standards and profiles that are in accord with ISO/IEC 9945 (POSIX).

This guide represents the POSIX OSE as it existed when the guide was approved.

2.2.2.31 POSIX OSE Cross-Category Services: A set of tools and/or features that has a direct effect on the operation of one or more component of the POSIX Open System Environment, but is not in and of itself a stand-alone component.

2.2.2.32 POSIX Standardized Profile (POSIX SP): A Standardized Profile that specifies the application of certain POSIX base standards in support of a class of applications and does not require any departure from the structure defined by the POSIX.0 Reference Model for POSIX systems.

NOTE: Which POSIX base standards form the basis of the POSIX SPs is still open. Annex A discusses some of the issues involved.

2.2.2.33 process: An address space and single thread of control that executes within that address space, and its required system resources.

A process is created by another process issuing the *fork*() function. The process that issues *fork*() is known as the parent process, and the new process created by the *fork*() as the child process.

2.2.2.34 profile: A set of one or more base standards, and, where applicable, the identification of chosen classes, subsets, options, and parameters of those base standards, necessary for accomplishing a particular function.

2.2.2.35 programming language API: The interface between applications and application platforms traditionally associated with programming language specifications, such as program control, math functions, string manipulation, etc.

2.2.2.36 protocol (OSI): A set of semantic and syntactic rules that determine the behavior of [OSI-] entities in the same layer in performing communication functions.

2.2.2.37 redirection: A system profile construction method of starting at a base platform and adding new services by allowing a service component to ask the base platform to redirect all requests for that type of service to the service component.

2.2.2.38 public specifications: Specifications that are available to anyone for implementation and distribution (i.e., sale) of that implementation without restriction.

2.2.2.39 reference model: A simplified description or representation of something.

2.2.2.40 scaleability: The ease with which software can be transferred from one graduated series of application platforms to another.
A simplified description or representation of something.

2.2.2.41 security: The protection of computer hardware and software from accidental or malicious access, use, modification, destruction, or disclosure.

Tools for the maintenance of security are focused on availability, confidentiality, and integrity.

2.2.2.42 service delivery latency: The interval between (a) context switch from an application context to the operating system context, and (b) satisfaction of the service request.

2.2.2.43 service request latency: The interval between (a) context switch from an application context to the operating system context, and (b) the reverse context switch from the operating system context to the application context for a given service request.

2.2.2.44 software: The programs, procedures, rules, and any associated documentation pertaining to the operation of a data processing system.

2.2.2.45 specification: A document that prescribes, in a complete, precise, verifiable manner, the requirements, design, behavior, or characteristics of a system or system component.

2.2.2.46 standardized profile: A balloted, formal, harmonized document that specifies a profile.

2.2.2.47 standards: Documents, established by consensus and approved by a recognized body, that provide, for common and repeated use, rules, guidelines, or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context.

NOTE: Standards should be based on the consolidated results of science, technology, and experience, and aimed at the promotion of optimum community benefits.

2.2.2.48 System Internal Interface (SII): The interface between application platform service components within that platform; it may be standardized or non-standard.

2.2.2.49 system services: Firmware and software that provide an aggregation of network element functions into a higher level function; provide an interface to the data contained in the system.

2.2.2.50 System Services API: An interface providing access to services associated with the application's internal resources.

The System Services API has two parts: Language Specifications and Processing Services API.

2.2.2.51 system software: Application-independent software that supports the running of application software.

2.2.2.52 transaction: A unit of work consisting of an arbitrary number of individual operations all of which will complete successfully (or be of no effect) on the intended resources.

A transaction has well defined boundaries. A transaction starts with a request from the application program and either completes successfully (commits) or has no effect (abort). Both the commit and abort signify a transaction completion.

2.2.2.53 transaction application program: A program written to meet the requirements of a chosen Transaction Processing (TP) application.

Such programs allow a sequence of operations that involve resources such as terminals and databases. The transaction AP specifies transaction boundaries. The transaction AP as defined here is a logical entity and may involve an arbitrary number of processes.

2.2.2.54 validation: The process of evaluating a ported application, software, or system to ensure compliance with requirements.

2.2.3 Abbreviations

For the purposes of this guide, the following abbreviations apply:

2.2.3.1 API: Application Program Interface

2.2.3.2 EEI: External Environment Interface