JTC1 and SC22 - Terminology

Background

Following my offer to collect together the definitions from SC22 standards, SC22 accepted my offer and appointed me as its terminology representative (I was later also asked to represent UK and BSI) on the JTC1 ad hoc group on terminology.

These notes summarise the results of collecting the SC22 definitions, and my impressions of the JTC1 ad hoc group.

Roger Scowen
August 2005

A collection of definitions from SC22 standards

SC22 asked me to prepare a collected terminology for SC22 containing the definitions from standards for which SC22 is responsible, and asked the project editors to send me the definitions in text form. Many, but not all, project editors did so. However there are sufficient for SC22 to judge whether to complete the list or abandon it as an interesting but unprofitable exercise.

Adding definitions to the database

The project editor of a standard typically sends the definitions from the standard as a Word file, but it may be plain text or in Latex or nroff format.

These definitions are transformed into a uniform format by a series of global 'find & replace' operations to produce a Word file where each definition is represented as a row of a table with three columns: the term, its definition, and any notes and/or examples. It is often easier to check this has been accomplished by copying and pasting successive attempts into Excel than examining the Word file itself.

Sometimes there are special cases such as exotic characters (for example Greek or mathematical characters), special fonts, illustrations, diagrams, or tables. There may also be cross references to other clauses in the standard, or URLs. It may be necessary to simplify or even remove these exceptions, but to note all these cases so that they can be restored in a published terminology.

When the definitions have been converted into a table, it is copied to an Excel worksheet. The clause numbers are now restored as two initial columns: one right-justified column for the constant element of the clause (for example '′3.', using Edit, Fill, Down), and a second left justified column for the varying clause number (using Edit, Fill, Series, increment 1). Another column is added to contain a mnemonic identifying the standard from which the definition is taken.

The next stage is to add the definitions for the standard to a worksheet containing all the definitions that have been collected, and to sort all the columns using the term as the key (Select all columns, Data, Sort, Sort by Column D).

Note that cross references in a definition (for example, indicated by words in italics in Prolog definitions) naturally refer to terms defined in the source standard.

The 2005 database

The 2005 database contains 1187 definitions collected from 12 standards and Technical Reports, in particular from Ada, Cobol, Extended BNF, ISLisp, Posix, Prolog, and WG11 (Language binding).

Sources

LISS	ISO/IEC TR 14369:1999 Language Independent Service Specifications
Prolog	ISO/IEC 13211-1:1995 Prolog: Part 1 - General core
Posix	ISO/IEC 9945-1:2003 POSIX: Part 1 - Base definitions
ISLisp	ISO/IEC 13816:1996 ISLisp
Cobol	ISO/IEC 1989:2002 Cobol
Extended BNF	ISO/IEC 14977:1996 Extended BNF
Ada	ISO/IEC 8652:1995/Cor.1:2000
ASIS	ISO/IEC FDIS 15291 Ada Semantic Interface Specification
ACATS	ISO/IEC 18009:1999 Ada Conformity Assessment Test Suite

Notes

ISO/IEC TR 14369 Language Independent Service Specifications	The definitions are copied from four other standards: TR 10182 (Guidelines for language bindings), 10967-1 (Language-independent arithmetic), 11404 (Language-independent datatypes), 13886 (Language-Independent Procedure Calling)
ISO/IEC TR 15942:2000	has no definitions clause.
ISO/IEC 18009	The definitions are mostly quoted from other sources, primarily ISO vocabulary guides and TR 9547. Only the definitions in clause 4.15.x are original to ACATS

Confession

The contents of the database have not been checked against the original standards, and I feel sure some errors have been introduced.

Files

The database has been provided as two Excel files, each with a single worksheet:

(1) collected sc22 definitions 2005.xls, and

(2) sorted sc22 definitions 2005.xls

Report of JTC1 ad hoc group on terminology meeting, 2005-05-02 to 04, Gatineau, Canada

JTC1 set up this ad hoc group following an offer by Canada to fill the gap in IT vocabulary standardization that has arisen since SC1, the relevant subcommittee, was closed about five years ago. The standard it produced and revised, ISO/IEC 2382, is now obsolete and there are no means to revise it.

Canada set up this meeting of the Ad Hoc Group partly to demonstrate the facilities and expertise of the Canada Translation Bureau (CTB) to the SC representatives on the Ad Hoc Group, and also to explain how they envisage the project proceeding.

The minutes and presentations made during the meeting are to be published by JTC1; what follows are my personal impressions and views.

Attenders

Only six SCs (subcommittees) and two NBs (National bodies) were present. The vast majority were Canadian Nationals: two from Europe, none from Asia. The convener claimed this indicates that most SCs have no problems or queries about the proposal. I believe it indicates apathy or lack of manpower.

Report

Requirement - An English and French database of specialist terminology is an essential resource in Canada where official (and many other) documents need to be published in both languages. Thus Canada needs ISO 2382 with its bilingual terminology of information technology - or something comparable with official status. Canada demonstrated the Termium system and software that the CTB uses to produce and maintain its database.

Canada's proposal - Canada envisages the JTC1 subcommittees updating the relevant (English) parts of ISO 2382, and providing the CTB with the results so that the CTB and SC together can produce and check the French equivalent terms. In each SC, there will be one representative who will liaise with the CTB and the rest of the SC.

Producing a subject terminology - Staff at the CTB currently use software called Termicom to develop and maintain a subject terminology, and offer to provide this to each SC representative. This is an interactive system for developing, examining and updating one term at a time, and thus loading a set of terms from a newly agreed standard will require batch input provided by the CTB. The approach planned by the CTB could thus require a great deal of work for the SC terminology representative, especially if liaison turns out to be back and forth with each of the WGs: this would onerous in many of the SCs.

A terminology tutorial - The CTB have produced an online training package, *The Pavel Terminology Tutorial*, to familiarize subject experts with the fundamental concepts of producing a terminology for a subject. English and French versions are currently available. I recommend project editors to look at

The Pavel Terminology Tutorial when they come to draft the definitions in their standard. It is accessible at `www.termium.com`.

Authority without control - ISO, JTC1, NBs, SCs, WGs, and Project editors all have different powers and responsibilities. Theoretically, there is a clear line of authority, control and management, but ultimately work on standardization is paid for by the employers of the WG members, and if they are unwilling to support work, then it is done slowly or not at all.

SC22 and ISO 2382 - JTC1 N6397 (2001-03-28) assigned responsibility to SC22 for maintenance of several parts of ISO 2382:

Part	Subject
2	Arithmetic and logic operations
7	Computer programming
15	Programming languages
16	Information theory

SC22 has so far ignored this responsibility.

SC22 and TCB experiment - I offered to provide a specimen of definitions from SC22 standards, work I have already volunteered to produce for SC22, in order to give CTB some idea of the scale of the project, and to see whether they can add value by identifying poor definitions and synonyms, and providing French translations. This project is under way with CTB examining the first 100 definitions from Prolog, Posix and LISS, i.e. terms starting with ‘a’ and ‘b’. A verbal report will be given at the plenary.

Query - How does the European Commission cope with its even greater translation needs?

Conclusions

(1) *The Pavel Terminology Tutorial*, freely accessible at `www.termium.com`, is a useful tool when starting to produce a terminology for a subject.

(2) SC22 should review its responsibilities concerning ISO 2382.

(3) SC22 and BSI should cautiously welcome the Canadian proposal even though several problems remain, because if they can be overcome, we will not have to behave like Humpty Dumpty, ‘When *I* use a word, it means just what I choose it to mean—neither more nor less’.

(4) What next? - Is it worth extending the 2005 list of definitions by including more SC22 standards? What have you learnt from the 2005 list? What use will it be? Should it be published as a Technical Report?

Document history

2005 May 13-30: File started and stored as c:\r5\standard\st82.doc.

2005-06-01: Emailed to SC22 and BSI.

2005-08-15: Extended into a report to SC22 plenary and stored in c:\my briefcase\sc22 – 2005 terminology.com.

Clause	Source	Term	Definition	Notes
B	LISS	abstract service interface	(ISO/IEC TR 10182) - An interface having an abstract definition that defines the format and the semantics of the function invoked independently of the concrete syntax (actual representation) of the values and the invocation mechanism.	
B	LISS	actual parameter	(ISO/IEC 13886) - A value that is bound to a formal parameter during the execution of a procedure.	
B	LISS	actual parametric datatype	(ISO/IEC 11404) - A datatype appearing as a parametric datatype in a use of a datatype generator, as opposed to the formal- parametric-types appearing in the definition of the datatype generator.	
B	LISS	actual parametric value	(ISO/IEC 11404) - A value appearing as a parametric value in a reference to a datatype family or datatype generator, as opposed to the formal-parametric-values appearing in the corresponding definitions.	
B	LISS	aggregate datatype	(ISO/IEC 11404) - A generated datatype each of whose values is made up of values of the component datatypes, in the sense that operations on all component values are meaningful.	
B	LISS	alien syntax	(ISO/IEC TR 10182) - Syntax of a language other than the host language.	
B	LISS	annotation	(ISO/IEC 11404) - A descriptive information unit attached to a datatype, or a component of a datatype, or a procedure (value), to characterize some aspect of the representations, variables, or operations associated with values of the datatype which goes beyond the scope of this International Standard.	
B	LISS	approximate	(ISO/IEC 11404) - A property of a datatype indicating that there is not a 1-to-1 relationship between values of the conceptual datatype and the values of a valid computational model of the datatype.	

Clause	Source	Term	Definition	Notes
B	LISS	arithmetic datatype	(ISO/IEC 10967-1) - A datatype whose values are members of Z, R, or C.	Note (original) - This standard specifies requirements for integer and floating point datatypes. Complex numbers are not covered by this standard, but will be included in a subsequent part of this standard.
B	LISS	association	(ISO/IEC 13886) - Any mapping from a set of symbols to values.	
B	LISS	axiom	(ISO/IEC 10967-1) - A general rule satisfied by an operation and all values of the datatype to which the operation belongs. As used in the specifications of operations, axioms are requirements.	
B	LISS	bounded	(ISO/IEC 11404) - A property of a datatype, meaning both bounded above and bounded below.	
B	LISS	bounded above	(ISO/IEC 11404) - A property of a datatype indicating that there is a value U in the value space such that, for all values s in the value space, s<=U.	
B	LISS	bounded below	(ISO/IEC 11404) - A property of a datatype indicating that there is a value L in the value space such that, for all values s in the value space, L<=s.	
B	LISS	box	(ISO/IEC 13886) - A model of a variable or container that holds a value of a particular type.	
B	LISS	characterizing operations	(ISO/IEC 11404) - (of a datatype) A collection of operations on, or yielding, values of the datatype, which distinguish this datatype from other datatypes with identical value spaces; (of a datatype generator) A collection of operations on, or yielding, values of any datatype resulting from an application of the datatype generator, which distinguish this datatype generator from other datatype generators which produce identical value spaces from identical parametric datatypes.	
B	LISS	client interface binding	(ISO/IEC 13886) - The possession by the client procedure of an interface reference.	
B	LISS	client procedure	(ISO/IEC 13886) - A sequence of instructions which invokes another procedure.	

Clause	Source	Term	Definition	Notes
B	LISS	complete procedure closure	(ISO/IEC 13886) - A procedure closure, all of whose global symbols are mapped.	
B	LISS	component datatype	(ISO/IEC 11404) - A datatype which is a parametric datatype to a datatype generator, i.e. a datatype on which the datatype generator operates.	
B	LISS	configuration	(ISO/IEC 13886) - Host and target computers, any operating system(s) and software used to operate a processor.	
B	LISS	continuation value	(ISO/IEC 10967-1) - A computational value used as the result of an arithmetic operation when an exception occurs. Continuation values are intended to be used in subsequent arithmetic processing. (Contrast with exceptional value).	Note (original) - The infinities and NaNs produced by an IEC 559 system are examples of continuation values. [CRLF] Note (additional) - Here "IEC 559 system" means a system conforming to IEC 559:1989 (IEEE 754:1985) Standard for binary floating-point arithmetic.
B	LISS	datatype	(ISO/IEC 10967-1) - A set of values and a set of operations that manipulate those values.	Notes (additional) [CLRF] 1. The purpose of the LIA-1 standard is to provide rigorous definitions of the basic arithmetic operations on integer and floating point datatype values. Hence, in the context of usage in that standard, the term "datatype" naturally includes the operations. [CLRF] 2. In the LIA standard, the first to be published, "datatype" is spelled with a space, i.e. "data type". (The same is true for the GLB Technical Report.) For consistency in this (LISS) Technical Report, "datatype" is substituted throughout.
B	LISS	datatype	(ISO/IEC 11404) - A set of distinct values, characterized by properties of those values and by operations on those values.	Note (additional) - This definition is essentially identical to that in this Technical Report, though emphasizing the "characterizing" role of operations in helping to identify corresponding LID datatypes to those in a particular language.

Clause	Source	Term	Definition	Notes
B	LISS	datatype	(ISO/IEC TR 14369) - A set of values, usually accompanied by a set of operations on those values.	
B	LISS	datatype declaration	(ISO/IEC 11404) - (1) The means provided by this International Standard for the definition of a language-independent datatype which is not itself defined by this International Standard; [CLRF] (2) An instance of use of this means.	
B	LISS	datatype family	(ISO/IEC 11404) - A collection of datatypes which have equivalent characterizing operations and relationships, but value spaces which differ in the number and identification of the individual values.	
B	LISS	datatype generator	(ISO/IEC 11404) - An operation on datatypes, as objects distinct from their values, which generates new datatypes.	
B	LISS	defined datatype	(ISO/IEC 11404) - A datatype defined by a type-declaration.	
B	LISS	defined generator	(ISO/IEC 11404) - A datatype generator defined by a type-declaration.	
B	LISS	denormalization loss	(ISO/IEC 10967-1) - A larger than normal rounding error caused by the fact that denormalized values has less than full precision. (See float-rounding for a full definition.)	
B	LISS	denormalized	(ISO/IEC 10967-1) - Those values of a floating point type F that provide less than the full precision allowed by that type.	
B	LISS	embedded alien syntax	(ISO/IEC TR 10182) - Statements in a special language for access to a system facility, included in a source program written in a standard programming language.	
B	LISS	error	(ISO/IEC 10967-1) - (1) The difference between a computed value and the correct value. (Used in phrases like "rounding error" or "error bound".) [CLRF] (2) A synonym for exception in phrases like "error message" or "error output". Error and exception are not synonyms in any other context.	

Clause	Source	Term	Definition	Notes
B	LISS	exact	(ISO/IEC 11404) - A property of a datatype indicating that every value of the conceptual datatype is distinct from all others in any valid computational model of the datatype.	
B	LISS	exception	(ISO/IEC 10967-1) - The inability of an operation to return a suitable numeric result. This might arise because no such result exists mathematically, or because the mathematical result cannot be represented with sufficient accuracy.	
B	LISS	exceptional value	(ISO/IEC 10967-1) - A non-numeric value produced by an arithmetic operation to indicate the occurrence of an exception. Exceptional values are not used in subsequent arithmetic processing.	Notes (original) [CLRF] 1. Exceptional values are used as part of the defining formalism only. With respect to this international standard, they do not represent values of any of the datatypes described. There is no requirement that they be represented or stored in the computing system. [CLRF] 2. Exceptional values are not to be confused with the NaNs and infinities defined in IEC 559. Contrast this definition with that of continuation value above.
B	LISS	execution sequence	(ISO/IEC 13886) - A succession of global states s1, s2, ... where each state beyond the first is derived from the preceding one by a single create operation or a single write operation.	
B	LISS	exponent bias	(ISO/IEC 10967-1) - A number added to the exponent of a floating point number, usually to transform the exponent to an unsigned integer.	
B	LISS	external identifier	(ISO/IEC TR 10182) - An identifier that is visible outside of a program.	
B	LISS	formal parameter	(ISO/IEC 13886) - The name symbol of a parameter used in the definition of a procedure to which a value will be bound during execution.	

Clause	Source	Term	Definition	Notes
B	LISS	formal-parametric-type	(ISO/IEC 11404) - An identifier, appearing in the definition of a datatype generator, for which a language-independent datatype will be substituted in any reference to a (defined) datatype resulting from the generator.	
B	LISS	formal-parametric-value	(ISO/IEC 11404) - An identifier, appearing in the definition of a datatype family or datatype generator, for which a value will be substituted in any reference to a (defined) datatype in the family or resulting from the generator.	
B	LISS	functional interface	(ISO/IEC TR 10182) - The abstract definition of the interface to a system facility by which system functions are provided.	
B	LISS	functional specification	(ISO/IEC TR 10182) - The specification of a system facility. In the context of this document, the functional specification is normally a potential or actual standard. For each system function the specification defines the parameters for invocation and their effects.	
B	LISS	generated datatype	(ISO/IEC 11404) - A datatype defined by the application of a datatype generator to one or more previously-defined datatypes.	
B	LISS	generated internal datatype	(ISO/IEC 11404) - A datatype defined by the application of a datatype generator defined in a particular programming language to one or more previously-defined internal datatypes.	
B	LISS	generator declaration	(ISO/IEC 11404) - (1) The means provided by this International Standard for the definition of a datatype generator which is not itself defined by this International Standard; [CLRF] (2) An instance of use of this means.	
B	LISS	global state	(ISO/IEC 13886) - The set of all existing boxes and their currently assigned values.	
B	LISS	global symbol	(ISO/IEC 13886) - Symbol used to refer to values that are permanently associated with a procedure.	

Clause	Source	Term	Definition	Notes
B	LISS	helper function	(ISO/IEC 10967-1) - A function used solely to aid in the expression of a requirement. Helper functions are not visible to the programmer, and are not required to be part of an implementation. However, some implementation-defined helper functions are required to be documented.	
B	LISS	host language	(ISO/IEC TR 10182) - The programming language for which a standard language binding is produced; the language in which a program is written.	
B	LISS	identifier	(ISO/IEC TR 10182) - Name of an object in an application program that uses a system facility.	
B	LISS	implementation (of this standard)	(ISO/IEC 10967-1) - The total arithmetic environment presented to a programmer, including hardware, language processors, exception handling facilities, subroutine libraries, other software, and all pertinent documentation.	
B	LISS	implementation-defined	(ISO/IEC TR 10182) - Possibly differing between different processors for the same language, but required by the language standard to be defined and documented by the implementor.	
B	LISS	implementation defined	(ISO/IEC 13886) - An implementation defined feature is a feature that is left implementation dependent by this International Standard, but any implementation claiming conformity to this standard shall explicitly specify how this feature is provided.	
B	LISS	implementation-dependent	(ISO/IEC TR 10182) - Possibly differing between different processors for the same language, and not necessarily defined for any particular processor.	
B	LISS	implementation dependent	(ISO/IEC 13886) - An implementation dependent feature is a feature that shall be provided by an implementation claiming conformity to this standard, but the implementation need not to specify how the feature is provided.	
B	LISS	implementor	(ISO/IEC TR 10182) - The individual or organization that realizes a system facility through software, providing access to the system functions by means of the standard language bindings.	

Clause	Source	Term	Definition	Notes
B	LISS	input parameter	(ISO/IEC 13886) - A formal parameter with an attribute indicating that the corresponding actual parameter is to be made available to the server procedure on entry from the client procedure.	
B	LISS	input/output parameter	(ISO/IEC 13886) - A formal parameter with an attribute indicating that the corresponding actual parameters are made available to the server procedure on entry from the client procedure and to the client procedure on return from the server procedure.	
B	LISS	interface	(ISO/IEC TR 14369) - The mechanism by which a service user invokes and makes use of a service.	
B	LISS	interface closure	(ISO/IEC 13886) - A collection of names and a collection of procedure closures, with a mapping between them.	
B	LISS	interface execution context	(ISO/IEC 13886) - The union of the procedure execution contexts for a given interface closure.	
B	LISS	interface reference	(ISO/IEC 13886) - An identifier that denotes a particular interface instance.	
B	LISS	interface type	(ISO/IEC 13886) - A collection of names and a collection of procedure types, with a mapping between them.	
B	LISS	interface type identifier	(ISO/IEC 13886) - An identifier that denotes an interface type.	
B	LISS	internal datatype	(ISO/IEC 11404) - A datatype whose syntax and semantics are defined by some other standard, language, product, service or other information processing entity.	
B	LISS	inward mapping	(ISO/IEC 11404) - A conceptual association between the internal datatypes of a language and the language-independent datatypes which assigns to each in datatype either a single semantically equivalent internal datatype or no equivalent internal datatype.	

Clause	Source	Term	Definition	Notes
B	LISS	invocation association	(ISO/IEC 13886) - The invocation association of a procedure closure <Image, Association> applied to a set of actual parameter values is the association of the closure augmented by a mapping of all local symbols to values and all formal parameter symbols to the corresponding actual parameter values. Thus it is a binding to values of all symbols in the procedure image for the duration of the invocation.	
B	LISS	invocation context	(ISO/IEC 13886) - For a particular procedure call, the instance of the objects referenced by the procedure, where the lifetime of the objects is bounded by the lifetime of the call.	
B	LISS	language	(ISO/IEC TR 14369) - Unless otherwise qualified, "language" means "programming language", not "specification language" or "natural (human) language".	
B	LISS	language binding	(ISO/IEC TR 14369) - A specification of the standard interface to a service, or set of services, for applications written in a particular programming language.	
B	LISS	language binding of F to L or L language binding of F	(ISO/IEC TR 10182) - A specification of the standard interface to facility F for programs written in language L.	
B	LISS	language committee	(ISO/IEC TR 10182) - The ISO technical subcommittee or working group responsible for the definition of a programming language standard.	
B	LISS	language-dependent	(ISO/IEC TR 14369) - Making use of the concepts, features or assumptions of a particular programming language.	
B	LISS	language-independent	(ISO/IEC TR 14369) - Not making use of the concepts, features or assumptions of any particular programming language or style of language.	
B	LISS	language-independent datatype	(ISO/IEC 11404) - (1) A datatype defined by this International Standard, or [CLRF] (2) A datatype defined by the means of datatype definition provided by this International Standard.	Note (additional) - The LID standard abbreviates this term to "LI datatype"

Clause	Source	Term	Definition	Notes
B	LISS	language processor	(ISO/IEC TR 14369) - The entire computing system which enables a programming language user to translate and execute programs written in the language, in general consisting both of hardware and of the relevant associated software.	
B	LISS	lower bound	(ISO/IEC 11404) - In a datatype which is bounded below, the value L such that, for all values s in the value space, L<=s.	
B	LISS	mapping	(ISO/IEC 11404) - (of datatypes) A formal specification of the relationship between the (internal) datatypes which are notions of, and specifiable in, a particular programming language and the (language-independent) datatypes specified in this International Standard; [CLRF] (of values) A corresponding specification of the relationships between values of the internal datatypes and values of the language-independent datatypes.	
B	LISS	mapping (in general)	(noun) - A defined association between elements (such as concepts, features or facilities) of one entity (such as a programming language, or a specification, or a standard) with corresponding elements of another entity. Mappings are usually defined as being from one entity into another. A language binding of a language L into a standard S usually incorporates both a mapping from L into S and a mapping from S into L. [CRLF] (verb) The process of determining or utilizing a mapping.	Note (additional) - This is essentially the same definition as for the LID standard, though necessarily made more general.
B	LISS	marshalling	(ISO/IEC 13886) - A process of collecting actual parameters, possibly converting them, and assembling them for transfer.	Note (additional) - The definition in this Technical Report is essentially identical, though spelled out more in the absence of the full context of the LIPC standard, and extended (in a Note) to preparing input values for a service.

Clause	Source	Term	Definition	Notes
B	LISS	marshalling	(ISO/IEC TR 14369) - The process of collecting the actual parameters used in a procedure call, converting them if necessary, and assembling them for transfer to the called procedure. This process is also carried out by the called procedure when preparing to return the results of the call to the caller.	
B	LISS	normalized	(ISO/IEC 10967-1) - Those values of a floating point type F that provide the full precision allowed by that type.	
B	LISS	notification	(ISO/IEC 10967-1) - The process by which a program (or that program's user) is informed that an arithmetic exception has occurred. For example, dividing 2 by 0 results in a notification.	
B	LISS	operation	(ISO/IEC 10967-1) - A function directly available to the user, as opposed to helper functions or theoretical mathematical functions.	
B	LISS	order	(ISO/IEC 11404) - A mathematical relationship among values.	Note (additional) - The LID standard also makes a cross-reference to its subclause 6.3.2.
B	LISS	ordered	(ISO/IEC 11404) - A property of a datatype which is determined by the existence and specification of an order relationship on its value space.	
B	LISS	output parameter	(ISO/IEC 13886) - A formal parameter with an attribute indicating that the corresponding actual parameter is to be made available to the client procedure on return from the server procedure.	
B	LISS	outward mapping	(ISO/IEC 11404) - A conceptual association between the internal datatypes of a language and the language-independent datatypes which identifies each internal datatype with a single semantically equivalent language- independent datatype.	
B	LISS	parameter	(ISO/IEC 13886) - A parameter is used to communicate a value from a client to a server procedure. The value supplied by the client is the actual parameter, the formal parameter is used to identify the received value in the server procedure.	

Clause	Source	Term	Definition	Notes
B	LISS	parametric datatype	(ISO/IEC 11404) - A datatype on which a datatype generator operates to produce a generated datatype.	
B	LISS	parametric value	(ISO/IEC 11404) - (1) A value which distinguishes one member of a datatype family from another, or [CLRF] (2) A value which is a parameter of a datatype or datatype generator defined by a type-declaration.	Note (additional) - In relation to type-declaration the LID standard also makes a cross-reference to its subclause 9.1.
B	LISS	partial procedure closure	(ISO/IEC 13886) - A procedure closure, some of whose global symbols are not mapped. Procedure closures may be complete, with all global symbols mapped, or partial with one or more global symbols not mapped.	
B	LISS	precision	(ISO/IEC 10967-1) - The number of digits in the fraction of a floating point number.	
B	LISS	primitive datatype	(ISO/IEC 11404) - An identifiable datatype that cannot be decomposed into other identifiable datatypes without loss of all semantics associated with the datatype.	
B	LISS	primitive internal datatype	(ISO/IEC 11404) - A datatype in a particular programming language whose values are not viewed as being constructed in any way from values of other datatypes in the language.	
B	LISS	procedural binding	(ISO/IEC TR 10182) - The definition of the interface to a system facility available to users of a standard programming language through procedure calls.	
B	LISS	procedural interface definition language	(ISO/IEC TR 10182) - A language for defining specific procedures for interfacing to a system facility as used, for example, in IS 8907 Database Language NDL.	
B	LISS	procedure	(ISO/IEC TR 10182) - A general term used in this document to cover a programming language concept which has different names in different programming languages - subroutine and function in Fortran, procedure and function in Pascal, etc. A procedure is a programming language dependent method for accessing one or more system functions from a program. A procedure has a name and a set of formal parameters with defined datatypes. Invoking a procedure transfers control to that procedure.	

Clause	Source	Term	Definition	Notes
B	LISS	procedure	(ISO/IEC 13886) - The procedure value.	
B	LISS	procedure	(ISO/IEC TR 14369) - In this Technical Report, the term "procedure" is used in the generic sense to cover both those (sometimes called subroutines) which do not return a value associated with the procedure name, and those (sometimes called functions) which do, and hence can be called from within expressions.	
B	LISS	procedure call	(ISO/IEC 13886) - The act of invoking a procedure.	
B	LISS	procedure closure	(ISO/IEC 13886) - A pair <procedure image, association> where the association defines the mapping for the image's global symbols and no others.	Note (original) - Procedure closures are the values of procedure type referred to in ISO/IEC 11404 - Language Independent Datatypes.
B	LISS	procedure execution context	(ISO/IEC 13886) - For a particular procedure, an instance of the objects satisfying the external references necessary to allow the procedure to operate, where these objects have a lifetime longer than a single call of that procedure.	
B	LISS	procedure image	(ISO/IEC 13886) - A representation of a value of a particular procedure type, which embodies a particular sequence of instructions to be performed when the procedure is called.	
B	LISS	procedure invocation	(ISO/IEC 13886) - The object which represents the triple: procedure image, execution context, and invocation context.	
B	LISS	procedure name	(ISO/IEC 13886) - The name of a procedure within an interface type definition.	
B	LISS	procedure return	(ISO/IEC 13886) - The act of return from the server procedure with a specific termination.	
B	LISS	procedure type	(ISO/IEC 13886) - The family of datatypes each of whose members is a collection of operations on values of other datatypes. Note, this is a different definition from procedure value.	
B	LISS	procedure value	(ISO/IEC 13886) - A closed sequence of instructions that is entered from, and returns control to, an external source.	

Clause	Source	Term	Definition	Notes
B	LISS	processor	(ISO/IEC TR 10182) - A system or mechanism that accepts a program as input, prepares it for execution, and executes the process so defined with data to produce results.	
B	LISS	processor	(ISO/IEC 13886) - A compiler or interpreter working in combination with a configuration.	
B	LISS	programming language extensions with native syntax or native syntax binding	(ISO/IEC TR 10182) - The functionality of the system facilities is incorporated into the host programming language so that the system functions appear as natural parts of the language. The compiler processes the language extensions and generates the appropriate calls to the system facility functions.	
B	LISS	representation	(ISO/IEC 11404) - (of a language-independent datatype) The mapping from the value space of the language-independent datatype to the value space of some internal datatype of a computer system, file system or communications environment; (of a value) The image of that value in the representation of the datatype.	
B	LISS	rounding	(ISO/IEC 10967-1) - The act of computing a representable final result for an operation that is close to the exact (but unrepresentable) result for that operation. Note that a suitable representable result may not exist.	
B	LISS	rounding function	(ISO/IEC 10967-1) - Any function rnd: R -> X (where X is a discrete subset of R) that maps each element of X to itself, and is monotonic non-decreasing. Formally, if x and y are in R, x in X => rnd(x) = x and x < y => rnd(x) <= rnd(y) Note that if u in R is between two adjacent values in X, rnd(u) selects one of those adjacent values.	
B	LISS	round to nearest	(ISO/IEC 10967-1) - The property of a rounding function rnd that when u in R is between two adjacent values in X, rnd(u) selects the one nearest u. If the adjacent values are equidistant from u, either may be chosen.	
B	LISS	round toward minus infinity	(ISO/IEC 10967-1) - The property of a rounding function rnd that when u in R is between two adjacent values in X, rnd(u) selects the one less than u.	

Clause	Source	Term	Definition	Notes
B	LISS	round toward zero	(ISO/IEC 10967-1) - The property of a rounding function rnd that when u in R is between two adjacent values in X, rnd(u) selects the one nearest 0.	
B	LISS	server procedure	(ISO/IEC 13886) - The procedure which is invoked by a procedure call.	
B	LISS	service	(ISO/IEC TR 14369) - A facility or set of facilities made available to service users through an interface.	
B	LISS	service provider	(ISO/IEC TR 14369) - A computer system or set of computer systems that implements a service and makes it available to service users.	
B	LISS	service user	(ISO/IEC TR 14369) - An application (typically a program in some language) which makes use of a service.	
B	LISS	shall	(ISO/IEC 10967-1) - A verbal form used to indicate requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted.	
B	LISS	should	(ISO/IEC 10967-1) - A verbal form used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that (in the negative form) a certain possibility is deprecated but not prohibited.	

Clause	Source	Term	Definition	Notes
B	LISS	signature (of a function or operation)	(ISO/IEC 10967-1) - A summary of information about an operation or function. A signature includes the operation name, the minimum set of inputs to the operation, and the maximum set of outputs from the operation (including exceptional values if any). The signature add_I: I x I -> I U {integer_overflow} states that the operation named add_I shall accept any pair of I values as input, and (when given such input) shall return either a single I value as its output or the exceptional value integer_overflow. A signature for an operation or function does not forbid the operation from accepting a wider range of inputs, nor does it guarantee that every value in the output range will actually be returned for some input. An operation given inputs outside the stipulated input range may produce results outside the stipulated output range.	
B	LISS	specification language	(ISO/IEC TR 14369) - A formal language for defining the semantics of a service or an interface precisely and without ambiguity.	
B	LISS	subtype	(ISO/IEC 11404) - A datatype derived from another datatype by restricting the value space to a subset whilst maintaining all characterizing operations.	
B	LISS	symbol	(ISO/IEC 13886) - A program entity used to refer to a value.	
B	LISS	system facility	(ISO/IEC TR 10182) - A coherent collection of services to be made available in some way to an application program. The system facility may be defined as a set of discrete system functions with an abstract service interface.	
B	LISS	system facility committee	(ISO/IEC TR 10182) - The ISO technical subcommittee or working group responsible for the development of the functional specification of a system facility.	

Clause	Source	Term	Definition	Notes
B	LISS	system function	(ISO/IEC TR 10182) - An individual component of a system facility, which normally has an identifying title and possibly some parameters. A system function's actions are defined by its relationships to other system functions in the same system facility.	
B	LISS	termination	(ISO/IEC 13886) - A predefined status related to the completion of a procedure call.	
B	LISS	unmarshalling	(ISO/IEC 13886) - The process of disassembling the transferred parameters, possibly converting them, for use by the server procedure on invocation or by the client procedure upon procedure return.	Note (additional) - The definition in this Technical Report is essentially identical, though spelled out more in the absence of the full context of the LIPC standard, and extended (in a Note) to receipt of input values by a service.
B	LISS	unmarshalling	(ISO/IEC TR 14369) - The process of receiving and disassembling transferred parameters, and converting them if necessary, to prepare the values for further use. This process is carried out by the called procedure on receipt of the actual parameters for the call, and by the caller on receipt of the returned results of the call.	
B	LISS	upper bound	(ISO/IEC 11404) - In a datatype which is bounded above, the value U such that, for all values s in the value space, s<=U.	
B	LISS	value	(ISO/IEC 13886) - The set Value contains all the values that might arise in a program execution.	
B	LISS	value space	(ISO/IEC 11404) - The set of values for a given datatype.	
B	LISS	variable	(ISO/IEC 11404) - A computational object to which a value of a particular datatype is associated at any given time; and to which different values of the same datatype may be associated at different times.	
B	LISS	Z	(ISO/IEC TR 14369) - (1) (mathematics, e.g. ISO/IEC 10967-1:1993) the complex numbers [CLRF] (2) (pronounced "zed") a formal specification language, see ISO/IEC WD 13568.	
3. 1	**Prolog**	**A:**	The set of *atoms* (see 6.1.2b, 7.1.4).	
3. 2	**Prolog**	**activation:**	The process of *executing* an *activator* .	

Clause	Source	Term	Definition	Notes
3. 3	**Prolog**	**activator:**	The result of preparing a *goal* for *execution* (see 7.7.3).	
3. 4	**Prolog**	**algorithm, Herbrand:**	See 3.85 --- **Herbrand algorithm**.	
3. 5	**Prolog**	**alias:**	An *atom* associated with an open *stream* (see 7.10.2.2). The standard input *stream* has the alias `user_input`, and the standard output *stream* has the alias `user_output`(see 7.10.2.3).	NOTE – A *stream* can have many aliases, but an *atom* can be the *alias* of at most one *stream* .
3. 6	**Prolog**	**anonymous variable:**	A *variable* (represented in a *term* or *Prolog text* by _ which differs from every other *variable* (and anonymous variable (see 6.1.2,6.4.3).	
3. 7	**Prolog**	**argument:**	A *term* which is associated with a *predication* or *compound term* .	
3. 8	**Prolog**	**arithmetic data type:**	A *data type* whose values are members of *ZZ or RR*.	
3. 9	**Prolog**	**arity:**	The number of *arguments* of a *compound term* . Syntactically, a non-negative integer associated with a *functor* or *predicate* .	
3. 10	**Prolog**	**assert, to:**	To assert a *clause* is to add it to the *user-defined procedure* in the *database* defined by the *predicate* of that *clause* .	NOTE – It is unnecessary for the *user-defined procedure* to already exist.
3. 11	**Prolog**	**associativity (of an operator):**	Property of being non-associative, right-associative, or left-associative (see 6.3.4, table 4).	
3. 12	**Prolog**	**atom:**	A basic object, denoted by an *identifier* (see 6.1.2b, 7.1.4).	
3. 13	**Prolog**	**atom, null:**	See 3.117 --- **null atom**.	
3. 14	**Prolog**	**atom, one-char:**	See 3.119 --- **one-char atom**.	
3. 15	**Prolog**	**atomic term:**	An *atom* or a *number* .	
3. 16	**Prolog**	**axiom:**	A rule satisfied by an operation and all values of the *data type* to which the operation belongs.	
3. 17	**Prolog**	**backtrack, to:**	To return to the *choicepoint* of the current *goal* in order to attempt to r*e-execute* it (see 7.7.8).	
3. 18	**Prolog**	**bias, exponent:**	See 3.68 --- **exponent bias**.	
3. 19	**Prolog**	**body:**	A *goal* , distinguished by its context as part of a *rule* (see 3.154).	
3. 20	**Prolog**	**bootstrapped (built-in predicate):**	Defined as a special case of a more general *built-in predicate* (see 8.1.5).	

Clause	Source	Term	Definition	Notes
3. 21	**Prolog**	**built-in predicate:**	A *procedure* whose *execution* is implemented by the *processor* (see 8).	
3. 22	**Prolog**	**byte:**	An integer in the range [0 .. 255] (see 7.1.2.1).	
3. 23	**Prolog**	**C:**	The set of *characters* (see 7.1.4.1).	
3. 24	**Prolog**	**callable term:**	An *atom* or a *compound term* .	
3. 25	**Prolog**	**CC:**	The set of character codes (see 7.1.2.2).	
3. 26	**Prolog**	**character:**	A member of *C* --- an *implementation defined* character set (see 6.5, 7.1.4.1).	
3. 27	**Prolog**	**character, quoted:**	See 3.144 --- **quoted character**.	
3. 28	**Prolog**	**character, unquoted:**	See 3.194 --- **unquoted character**.	
3. 29	**Prolog**	**character-conversion mapping:**	A *mapping* on the set of *characters* , *C* , which specifies that, in some *Prolog text* units and *sources* , some *characters* are intended to be equivalent to other *characters* , and *converted* to those *characters* (see 3.46, 7.4.2.5, 8.14.5).	
3. 30	**Prolog**	**choicepoint:**	A state during *execution* from which a *goal* can be *executed* in more than one way.	
3. 31	**Prolog**	**class (of an operator):**	The class of an *operator* defines whether it is a prefix, infix, or postfix *operator* (see 6.3.4).	
3. 32	**Prolog**	**clause:**	A *fact* or a *rule* . It has two parts: a *head* , and a *body* .	NOTE – In ISO/IEC International Standards “clause” has the meaning: one of the numbered paragraphs of a standard. In this part of ISO/IEC 13211, the context distinguishes the two meanings.
3. 33	**Prolog**	**clause-term:**	A *read-term* `T.` in *Prolog text* where `T` does not have *principal functor* `(:-)/1` (see 6.2.1.2).	
3. 34	**Prolog**	**collating sequence:**	An *implementation defined* ordering defined on the set *C* of *characters* (see 6.6).	
3. 35	**Prolog**	**complete database:**	The set of *procedures* with respect to which *execution* is performed (see 7.5).	
3. 36	**Prolog**	**composition (of two substitutions:**	The mapping resulting from the application of the first *substitution* followed by the application of the second. Composition of the *substitutions* §1 and §2 is denoted §1 ° §2. When the composition acts on a term *t* , it is denoted by *t* §1 ° §2, with the meaning ((*t* §1) §2).	

Clause	Source	Term	Definition	Notes
3. 37	**Prolog**	**compound term:**	A *functor* of *arity* `N`, `N` positive, together with a sequence of `N` arguments (see 6.1.2e, 7.1.5)	
3. 38	**Prolog**	**configuration:**	Host and target computers, any operating system(s) and software used to operate a *processor* .	
3. 39	**Prolog**	**conforming processor:**	A *processor* which conforms to all the compliance clauses (see 5.1) for *processors* in this part of ISO/IEC 13211.	
3. 40	**Prolog**	**conforming Prolog data:**	Sequences of *characters* and *bytes* that conform to all the compliance clauses for *Prolog data* in this part of ISO/IEC 13211 (see 5, 6.2.2).	
3. 41	**Prolog**	**conforming Prolog text:**	A sequence of *characters* that conforms to all the compliance clauses for *Prolog text* in this part of ISO/IEC 13211 (see 5, 6.2).	
3. 42	**Prolog**	**construct, control:**	See 3.45 --- **control construct**.	
3. 43	**Prolog**	**constructor, list:**	See 3.100 --- **list constructor**.	
3. 44	**Prolog**	**contain, to:**	A *term* `T1` contains another *term* `T2` if either `T1` and `T2` are *identical terms* , or `T1` is a *compound term* , one of whose *arguments* contains `T2`.	
3. 45	**Prolog**	**control construct:**	A procedure whose definition is part of the Prolog *processor* (see7.8).	

Clause	Source	Term	Definition	Notes
3. 46	**Prolog**	**$Conv_C$:**	The *character-conversion mapping* on *C* (the set of *characters*) which specifies that, in some *Prolog text* units and *sources* , some *characters* are *converted* to other *characters* (see 3.29, 7.4.2.5, 8.14.5). The initial value of $Conv_C$ shall be $identity\_mapping_C$.	NOTES [CRLF] 1 A directive or goal `char_conversion(In, Out)` (7.4.2.5, 8.14.5) replaces $Conv_C$ by $update\_mapping_C$ (`In`, `Out`, $Conv_C$). [CRLF] 2 Any unquoted character `C` that is part of a read-term which is input by `read_term/3` (8.14.1) or as Prolog text is replaced by $apply\_mapping_C$ (`C`, $Conv_C$). [CRLF] 3 $Conv_C$ can be inspected by calling `current_char_conversion/2` (8.14.6). [CRLF] 4 The rationale for providing this facility is because some extended character sets (for example, Japanese JIS charactersets) are used with the basic character set and contain the characters equivalent to those in the basic character set with different encoding. In such cases, users will often wish the meaning of characters in Prolog data and Prolog text to be the same regardless of the encoding.
3. 47	**Prolog**	**convert (from type A to type B):**	An operation whose *signature* is [CRLF] $convert_{A \to B}$:A→ B ∪ {**error**} [CRLF] which value of *type A* to *typeB* . It shall be an error if the conversion cannot be made. For example, see converting a *term* to a *clause* and vice versa (7.6), *character-conversion* (3.29, 7.4.2.5, 8.14.5), and converting a *floating point value* to an *integer value* and vice versa (9.1.6).	
3. 48	**Prolog**	**copy, renamed (of a term):**	See 3.150 --- **renamed copy (of a term).**	
3. 49	**Prolog**	**CT:**	The set of *compound terms* (see 6.1.2e, 7.1.5).	
3. 50	**Prolog**	**cut:**	A *control construct* whose effect is to remove all *choicepoints* back to a deeper execution state defined by its cutparent (see 7.7.2, 7.8.4).	
3. 51	**Prolog**	**data, conforming Prolog:**	See 3.40 --- **conforming Prolog data**.	
3. 52	**Prolog**	**database:**	The set of *user-defined procedures* which currently exist during *execution* (see 7.5).	
3. 53	**Prolog**	**database, complete:**	See 3.35 **--- complete database.**	

Clause	Source	Term	Definition	Notes
3. 54	**Prolog**	**data type:**	A set of values and a set of operations that manipulate those values.	
3. 55	**Prolog**	**data type, arithmetic:**	See 3.8 --- **arithmetic data type**.	
3. 56	**Prolog**	**denormalized value:**	A *floating point value* of type F providing less than the full precision allowed by F (see F_D , 7.1.3).	
3. 57	**Prolog**	**directive:**	A *term* `D` which affects the meaning of *Prolog text* (see 7.4.2), and is denoted in that *Prolog text* by a *directive-term* `:-(D).`	
3. 58	**Prolog**	**directive-term:**	A *read-term* `T.` in *Prolog text* where `T` has *principal functor* `(:-)/1` (see 6.2.1.1).	
3. 59	**Prolog**	**dynamic (of a procedure):**	A dynamic *procedure* is one whose *clauses* can be inspected or altered during execution, for example by *asserting* or *retracting* * *clauses* (see 7.5.2).	
3. 60	**Prolog**	**effect, side:**	3.157 --- **side effect**.	
3. 61	**Prolog**	**element (of a list):**	An element of a *non-empty list* is either the *head* of the *list* or an element of the *tail* of the *list* . The *empty list* has no elements.	
3. 62	**Prolog**	**empty list:**	The *atom* `[]` (nil).	
3. 63	**Prolog**	**error:**	A special circumstance which causes the normal process of *execution* to be interrupted (see 7.12).	
3. 64	**Prolog**	**evaluable functor:**	The *principal functor* of an *expression* (see 7.9, 9).	
3. 65	**Prolog**	**evaluate:**	To reduce an *expression* to its value. (see 7.9, 8.6.1. 9).	
3. 66	**Prolog**	**exceptional value:**	A non-numeric value of an *expression* : **float_overflow, int_overflow, underflow, zero_divisor,** or **undefined** (see 7.9).	NOTE – It is an `evaluation_error(E)` when the value of an *expression* is an exceptional value.
3. 67	**Prolog**	**execution (verb: to execute):**	The process by which a Prolog *processor* tries to *satisfy* a *goal* (see 7.7).	
3. 68	**Prolog**	**exponent bias:**	A number added to the exponent of a floating point number, usually to convert the exponent to an unsigned integer.	
3. 69	**Prolog**	**expression:**	An *atomic term* or a *compound term* which may be *evaluated* to produce a value (see 8.6.1, 9).	

Clause	Source	Term	Definition	Notes
3. 70	**Prolog**	**extension:**	A facility provided by the *processor* that is not specified in this part of ISO/IEC 13211 but that would not cause any ambiguity or contradiction if added to this part of ISO/IEC 13211.	
3. 71	**Prolog**	**F:**	The set of floating point values (see 6.1.2d, 7.1.3).	
3. 72	**Prolog**	**fact:**	A *clause* whose *body* is the *goal* true.	NOTE – A fact can be represented in *Prolog text* by a *term* whose *principal functor* is neither (:-)/1 nor (:-)/2.
3. 73	**Prolog**	**fail, to:**	*Execution* of a *goal* fails if it is not *satisfied* .	
3. 74	**Prolog**	**file name:**	An *implementation defined* * *ground term* which identifies to the *processor* a file which will be used for input/output during the *execution* of the *Prolog text* .	
3. 75	**Prolog**	**flag:**	An *atom* which is associated with an *implementation defined* or user-defined value (see 7.11).	
3. 76	**Prolog**	**floating point value:**	A member of the set *F* (see 6.1.2d, 7.1.3).	
3. 77	**Prolog**	**functor:**	An *identifier* together with an *arity* .	
3. 78	**Prolog**	**functor name:**	The *identifier* of a *functor* .	
3. 79	**Prolog**	**function, rounding:**	See 3.153 --- **rounding function**.	
3. 80	**Prolog**	**functor, principal:**	See 3.134 --- **principal functor**.	
3. 81	**Prolog**	**goal:**	A *predication* which is to be *executed* (see *body* , *query* , and 7.7.3).	
3. 82	**Prolog**	**ground term:**	An *atomic term* or a *compound term* whose *arguments* are all ground. A *term* is ground with respect to a *substitution* if application of the *substitution* yields a ground term.	
3. 83	**Prolog**	**head (of a list):**	The first *argument* of a *non-empty list* .	
3. 84	**Prolog**	**head (of a rule):**	A *predication* , distinguished by its context.	
3. 85	**Prolog**	**Herbrand algorithm:**	An algorithm which computes the *most general unifier MGU* of a set of equations (see 7.3.2).	
3. 86	**Prolog**	**I:**	The set of integers (see 6.1.2c, 7.1.2).	
3. 87	**Prolog**	**identical terms:**	Two *terms* are identical if they have the same abstract syntax (see 6.1.2).	
3. 88	**Prolog**	**identifier:**	A basic unstructured object used to denote an *atom* , *functor name* or *predicate name* .	
3. 89	**Prolog**	**iff:**	If and only if.	

Clause	Source	Term	Definition	Notes
3. 90	**Prolog**	**implementation defined:**	Defined partly by this part of ISO/IEC 13211, and partly by the documentation accompanying a *processor* (see 5).	
3. 91	**Prolog**	**implementation dependent:**	An implementation dependent feature is dependent on the *processor* .	NOTE – This part of ISO/IEC 13211 does not require an implementation dependent feature to be defined in the accompanying *processor* documentation.
3. 92	**Prolog**	**implementation specific:**	*Undefined* by this part of ISO/IEC 13211 but supported by a *conforming processor* .	NOTE – This part of ISO/IEC 13211 does not require an implementation specific feature to be supported by a conforming processor, but it preserves the syntax and semantics of a strictly conforming Prolog text which does not use it, for example, defining a term order on variables, or defining unification for terms which are *STO* (3.165).
3. 93	**Prolog**	**indicator, predicate:**	See 3.131 --- { predicate indicator.	
3. 94	**Prolog**	**input/output mode:**	An *atom* which represents an attribute of a *stream* . A *processor* shall support the *input/output modes* : `read`, `write`, `append` (see 8.11.5, 7.10.1.1).	
3. 95	**Prolog**	**instance (of a term):**	The result of applying a *substitution* to the *term* . If *t* is a term and § a *substitution* , the instance of *t* by § is denoted *t* §.	
3. 96	**Prolog**	**instantiated:**	A *variable* is instantiated with respect to a *substitution* if application of the *substitution* yields an *atomic term* or a com*pound term* . A *term* is instantiated if any of its *variables* are instantiated.	
3. 97	**Prolog**	**integer value:**	A member of the set *I* (see 6.1.2c, 7.1.2).	
3. 98	**Prolog**	**level, top:**	See 3.185 --- **top level**.	
3. 99	**Prolog**	**list:**	Either the *empty list* or a *non-empty list* .	NOTE – Examples: `[]`, `[a, X]`, `[1, 2, _]`, `[a | [b]]`
3. 100	**Prolog**	**list constructor:**	The *principal functor* '.'/2 used for constructing *lists* .	
3. 101	**Prolog**	**list, empty:**	See 3.62 --- **empty list**.	
3. 102	**Prolog**	**list, non-empty:**	See 3.114 --- **non-empty list**.	
3. 103	**Prolog**	**list, partial:**	See 3.125 --- **partial list**.	
3. 104	**Prolog**	**list, read-options:**	See 3.147 --- **read-options list**.	
3. 105	**Prolog**	**list, write-options:**	See 3.207 --- **write-options list**.	

Clause	Source	Term	Definition	Notes
3. 106	**Prolog**	**mapping:**	A *data type* M_T where T is a data type (see 4.3).	
3. 107	**Prolog**	**mode, input/output:**	See 3.94 --- **input/output mode**.	
3. 108	**Prolog**	**most general unifier (MGU):**	The most general unifier (*MGU*) of *terms* is a minimal *substitution* which acts on the *terms* to make them *identical*. Any unifier is an instance of some *MGU*.	NOTE – It is defined up to a renaming of the *variables*. If idempotent no *variable* of its domain appears in the resulting *terms*. An idempotent *MGU* can be computed by the *Herbrand algorithm* (see 7.3.2).
3. 109	**Prolog**	**name (of atom):**	A sequence of *characters* which distinguishes an *atom* from any different *atom* (see 6.1.2b).	
3. 110	**Prolog**	**name, file:**	See 3.74 --- **file name**.	
3. 111	**Prolog**	**name, functor:**	See 3.78 --- **functor name**.	
3. 112	**Prolog**	**name, predicate:**	See 3.132 --- **predicate name**.	
3. 113	**Prolog**	**named variable:**	A *variable* which is not an *anonymous variable* (see 6.1.2a, 6.4.3).	
3. 114	**Prolog**	**non-empty list:**	A *compound term* whose *principal functor* is the *list constructor* and whose second *argument* is a *list*.	
3. 115	**Prolog**	**normalized value:**	A *floating point value* of type F providing the full precision allowed by F (see 7.1.3).	
3. 116	**Prolog**	**NSTO:**	Not subject to occurs-check (see 7.3.3).	
3. 117	**Prolog**	**null atom:**	The *atom* `''`.	
3. 118	**Prolog**	**number:**	An *integer value* or *floating point value*.	
3. 119	**Prolog**	**one-char atom:**	An *atom* whose name is a single *character*.	
3. 120	**Prolog**	**operand (of a compound term or predication):**	An *argument* of a *compound term* (*predication*) whose *functor name* (*predicate name*) is an *operator*.	
3. 121	**Prolog**	**operand (of an operation):**	A value supplied to an operation defined by a *signature* and one or more *axioms*.	
3. 122	**Prolog**	**operator:**	A *functor name* or *predicate name* which allows *compound terms* or *predications* respectively, to be expressed in prefix, infix or postfix form (see 6.3.4).	
3. 123	**Prolog**	**operator, predefined:**	See 3.128 --- **predefined operator**.	
3. 124	**Prolog**	**options, stream:**	See 3.167 --- **stream-options**.	
3. 125	**Prolog**	**partial list:**	A *variable*, or a *compound term* whose *principal functor* is the *list constructor* and whose second *argument* is a partial list.	NOTE – The concept of a partial list is used in 8.5.3. Examples: `A`, `[a | X]`, `[1, 2 | B]`

Clause	Source	Term	Definition	Notes
3. 126	**Prolog**	**position, stream:**	See 3.168 --- **stream position**.	
3. 127	**Prolog**	**precision:**	The number of digits in the fraction of a *floating point value* (see 7.1.3).	
3. 128	**Prolog**	**predefined operator:**	An *operator* which is initially provided by the *processor* .	
3. 129	**Prolog**	**predicate:**	An *identifier* together with an *arity* .	
3. 130	**Prolog**	**predicate, built-in:**	See 3.21 --- **built-in predicate**.	
3. 131	**Prolog**	**predicate indicator:**	A *compound term* `A/N`, where `A` is an *atom* and `N` is a non-negative integer, denoting one particular *procedure* (see 7.1.6.6).	
3. 132	**Prolog**	**predicate name:**	The *identifier* of a *predicate* .	
3. 133	**Prolog**	**predication:**	A *predicate* with *arity* `N` and a sequence of `N` *arguments* .	
3. 134	**Prolog**	**principal functor:**	The principal functor of a *compound term* is `F/N` if the *functor* of the *compound term* is `F` and its *arity* is `N`. The principal functor of an *atomic term* is `C/0` if the *atomic term* is `C`.	
3. 135	**Prolog**	**private (of a procedure):**	A private *procedure* is one whose *clauses* cannot be inspected during *execution* . (see 7.5.3).	
3. 136	**Prolog**	**procedure:**	A *control construct* , a *built-in predicate* , or a *user-defined procedure* . A procedure is either *static* or *dynamic* . A procedure is either p*r* ivate or *public* (see 7.5).	
3. 137	**Prolog**	**procedure, user-defined:**	See 3.195 --- **user-defined procedure**.	
3. 138	**Prolog**	**processor:**	A compiler or interpreter working in combination with a *configuration* .	
3. 139	**Prolog**	**processor, conforming:**	See 3.39 --- **conforming processor**.	
3. 140	**Prolog**	**Prolog data:**	A sequence of *read-terms* (see 6.2.2).	
3. 141	**Prolog**	**Prolog text:**	A sequence of *read-terms* denoting *directives* and *clauses* (see 6.2, 7.4).	
3. 142	**Prolog**	**public (of a procedure):**	A public *procedure* is one whose *clauses* can be inspected during *execution* , for example by calling the *built-in predicate* : `clause/2` (see 7.5.3, 8.8.1).	

Clause	Source	Term	Definition	Notes
3. 143	**Prolog**	**query:**	A *goal* given as interactive input to the *top level*.	NOTE – This part of ISO/IEC 13211 does not define or require a *processor* to support the concept of *top level*.
3. 144	**Prolog**	**quoted character:**	A *character* in *Prolog text* or *Prolog data* which is a *single quoted character* or a *double quoted character* or a *back quoted character* (see 6.4.2.1).	NOTE – For example, `'a''b\'c'` contains 5 quoted characters (1) `a`, (2) `'`, (3) `b`, (4) `'` (a meta escape sequence), (5) `c`.
3. 145	**Prolog**	**R:**	The set of real numbers (see 4.1.1).	
3. 146	**Prolog**	**read-option:**	A *compound term* with *uninstantiated* * *arguments* which amplifies the results produced by the *built-in predicate* `read_term/3` (8.14.1) and the *bootstrapped* * *built-in predicates* based on it (see 7.10.3).	
3. 147	**Prolog**	**read-options list:**	A list of *read-options*.	
3. 148	**Prolog**	**read-term:**	A *term* followed by an end token (see 6.2.2, 6.4.8).	
3. 149	**Prolog**	**re-execute, to:**	To re-execute a *goal* is to attempt to *satisfy* it again (see 7.7.6, 7.7.8).	
3. 150	**Prolog**	**renamed copy (of a term)**	A special *variant* of a *term* (see 7.1.6.2).	
3. 151	**Prolog**	**retract, to:**	To retract a *clause* is to remove it from the *user-defined procedure* in the *database* defined by the *predicate* of that *clause*.	
3. 152	**Prolog**	**rounding:**	Computing a representable final value (for an operation) which is close to the exact (but unrepresentable) value for that operation (see 9.1.3.1, 9.1.4.1).	
3. 153	**Prolog**	**rounding function:**	A function with *signature* : *rnd* : $R \rightarrow X$ [CRLF] (where X is a discrete subset of R) which maps each member of X to itself, and is monotonic non-decreasing. Formally, if x and y are in R, [CRLF] $x \in X \Rightarrow rnd(x) = x$ [CRLF] $x < y \Rightarrow rnd(x) < rnd(y)$	NOTE – If $u \in R$ is between two adjacent values in X, *rnd(u)* selects one of those adjacent values.
3. 154	**Prolog**	**rule:**	A *clause* whose *body* is not the *goal* `true`. During *execution*, if the *body* is true for some *substitution*, then the *head* is also true for that *substitution*. A rule is represented in *Prolog text* by a *term* whose *principal functor* is `(:-)/2` where the first *argument* is *converted* to the *head*, and the second *argument* is *converted* to the *body*.	

Clause	Source	Term	Definition	Notes
3. 155	**Prolog**	**satisfy, to:**	To satisfy a *goal* is to *execute* it successfully.	
3. 156	**Prolog**	**sequence, collating:**	See 3.34 --- **collating sequence**.	
3. 157	**Prolog**	**side effect:**	A non-logical effect of an *activator* during *execution* (see 7.7.9).	
3. 158	**Prolog**	**signature:**	A specification of an *operation* which defines its name, and the *type* of its *operands(s)* and value.	NOTE – The operation is further defined by one or more *axioms* . For example, the signature: [CRLF] $add_I : I \times I \rightarrow I \cup$ {**int_overflow**} [CRLF] defines the *operation* add_I which takes two integer operands $I \times I$ and produces either a single *integer value* (I) or the *exceptional value* **int_overflow**.
3. 159	**Prolog**	**sink:**	A physical object to which a *processor* outputs results, for example a file, terminal, or interprocess communication channel (see 7.10.1).	
3. 160	**Prolog**	**source:**	A physical object from which a *processor* inputs data, for example a file, terminal, or interprocess communication channel (see 7.10.1).	
3. 161	**Prolog**	**source/sink:**	A *source* or a *sink* .	
3. 162	**Prolog**	**specifier (of an operator):**	One of the *atoms* `fx`, `fy`, `xfx`, `xfy`, `yfx`, `xf` or `yf`. A specifier denotes the *class* and *associativity* of an *operator* (see 6.3.4).	
3. 163	**Prolog**	**stack:**	A *data type* S_D where D is a *data type* (see 4.2).	
3. 164	**Prolog**	**static (of a procedure):**	A static *procedure* is one whose *clauses* cannot be altered (see 7.5.2).	
3. 165	**Prolog**	**STO:**	Subject to occurs-check (see 7.3.3).	
3. 166	**Prolog**	**stream:**	A connection to a *source* or *sink* (see 7.10.2).	
3. 167	**Prolog**	**stream-options:**	A list of zero or more *terms* which specify additional characteristics over and above those given by the *mode* of a *stream* (see 7.10.2.11).	
3. 168	**Prolog**	**stream position:**	An absolute position in a *source* /*sink* to which the *stream* is connected (see 7.10.2.8).	
3. 169	**Prolog**	**stream, target:**	See 7.10.2.5 --- **Target stream**.	
3. 170	**Prolog**	**stream-term:**	An *implementation dependent* * *ground term* which identifies a *stream* inside *Prolog text* (see 7.10.2.1).	

Clause	Source	Term	Definition	Notes
3. 171	**Prolog**	**substitution:**	A mapping from *variables* to *terms* . By extension a substitution acts on a *term* by acting on each *variable* in the *term* .	NOTE – A substitution is represented by a Greek letter (for example Σ, σ, μ) acting as a postfix *operator* , for example:
3. 172	**Prolog**	**succeed, to:**	*Execution* of a *goal* succeeds if it is *satisfied* .	
3. 173	**Prolog**	**tail:**	The second *argument* of a *non-empty list* .	
3. 174	**Prolog**	**target stream:**	See 7.10.2.5 --- **Target stream**.	
3. 175	**Prolog**	**term:**	An *atomic term* , a *compound term* or a *variable* (see 7.1).	
3. 176	**Prolog**	**term, atomic:**	See 3.15 --- **atomic term**.	
3. 177	**Prolog**	**term, callable:**	See 3.24 --- **callable term**.	
3. 178	**Prolog**	**term, compound:**	See 3.37 --- **compound term**.	
3. 179	**Prolog**	**term, ground:**	See 3.82 --- **ground term**.	
3. 180	**Prolog**	**terms, identical:**	See 3.87 --- **identical terms**.	
3. 181	**Prolog**	**term-precedes:**	A binary relation on the set of terms which defines a total ordering of terms (see 7.2).	
3. 182	**Prolog**	**term, stream:**	See 3.170 --- **stream-term**.	
3. 183	**Prolog**	**text, conforming Prolog:**	See 3.41 --- **conforming Prolog text**.	
3. 184	**Prolog**	**text, Prolog:**	See 3.141 --- **Prolog text**.	
3. 185	**Prolog**	**top level:**	A process whereby a Prolog *processor* repeatedly inputs and *executes* * *queries* .	NOTE – This part of ISO/IEC 13211 does not define or require a *processor* to support the concept of top level.
3. 186	**Prolog**	**type:**	The type of a *term* is a property of the term depending on its syntax and is one of: atom, integer, floating point, variable or compound term (see 7.1).	
3. 187	**Prolog**	**type, data:**	See 3.54 --- **data type**.	
3. 188	**Prolog**	**undefined:**	A feature is undefined if this part of ISO/IEC 13211 (1) states it is undefined, or (2) makes no requirements concerning its *execution* .	
3. 189	**Prolog**	**unifiable:**	Two or more *terms* are unifiable *iff* there exists a *unifier* for them.	
3. 190	**Prolog**	**unifier (of two or more terms):**	A *substitution* such that applying this *substitution* to each *term* results in *identical* terms.	
3. 191	**Prolog**	**unifier, most general:**	See 3.108 --- **most general unifier**.	

Clause	Source	Term	Definition	Notes
3. 192	**Prolog**	**unify, to:**	To find and apply a *most general unifier* of two *terms* by successfully executing (explicitly or implicitly) the *built-in predicate* `(=)/2` (unify) (see 8.2.1).	
3. 193	**Prolog**	**uninstantiated:**	A var iable is *uninstantiated* when it is not *instantiated* .	
3. 194	**Prolog**	**unquoted character:**	A *character* in *Prolog text* or *Prolog data* which is not a *quoted character* (see 6.4.2.1).	
3. 195	**Prolog**	**user-defined procedure:**	A *procedure* which is defined by a sequence of *clauses* where the *head* of each *clause* has the same *predicate indicator* , and each *clause* is expressed by *Prolog text* or has been *asserted* during *execution* (see 8.9).	
3. 196	**Prolog**	**V:**	The set of *variables* , (see 6.1.2a, 7.1.1).	
3. 197	**Prolog**	**value, denormalized:**	See 3.56 --- **denormalized value**.	
3. 198	**Prolog**	**value, exceptional:**	See 3.66 --- **exceptional value**.	
3. 199	**Prolog**	**value, normalized:**	See 3.115 --- **normalized value**.	
3. 200	**Prolog**	**variable:**	An object which may become *instantiated* to a *term* during *execution* . (see 6.1.2a, 7.1.1).	
3. 201	**Prolog**	**variable, anonymous:**	See **3.6 – anonymous variable**.	
3. 202	**Prolog**	**variable, named:**	See 3.113 --- **named variable**.	
3. 203	**Prolog**	**variable set (of a term):**	See 7.1.1.1 --- **Variable set of a term**.	
3. 204	**Prolog**	**variant (of a term):**	See 7.1.6.1 --- **Variants of a term**.	
3. 205	**Prolog**	**witness (of a set of variables):**	See 7.1.1.2 --- **Witness of a variable set**.	
3. 206	**Prolog**	**write-option:**	A *ground term* that controls the output produced by the *built-in predicate* `write_term/3` (8.14.2) and its *bootstrapped* * *built-in predicates* (see 7.10.4, 7.1.4.2).	
3. 207	**Prolog**	**write-options list:**	A list of *write-options* .	
3. 208	**Prolog**	**Z:**	The set of mathematical integers (see 4.1.1).	
3.	POSIX	Definitions	For the purposes of IEEE Std 1003.1-2001, the terms and definitions given in Definitions apply.	**Note:** No shading to denote extensions or options occurs in this chapter. Where the terms and definitions given in this chapter are used elsewhere in text related to extensions and options, they are shaded as appropriate.

Clause	Source	Term	Definition	Notes
3. 1	POSIX	Abortive Release	An abrupt termination of a network connection that may result in the loss of data.	
3. 2	POSIX	Absolute Pathname	A pathname beginning with a single or more than two slashes; see also Pathname.	Note: Pathname Resolution is defined in detail in Pathname Resolution.
3. 3	POSIX	Access Mode	A particular form of access permitted to a file.	
3. 4	POSIX	Additional File Access Control Mechanism	An implementation-defined mechanism that is layered upon the access control mechanisms defined here, but which do not grant permissions beyond those defined herein, although they may further restrict them.	Note: File Access Permissions are defined in detail in File Access Permissions.
3. 5	POSIX	Address Space	The memory locations that can be referenced by a process or the threads of a process.	
3. 6	POSIX	Advisory Information	An interface that advises the implementation on (portable) application behavior so that it can optimize the system.	
3. 7	POSIX	Affirmative Response	An input string that matches one of the responses acceptable to the *LC_MESSAGES* category keyword **yesexpr**, matching an extended regular expression in the current locale.	Note: The LC_MESSAGES category is defined in detail in LC_MESSAGES.
3. 8	POSIX	Alert	To cause the user's terminal to give some audible or visual indication that an error or some other event has occurred. When the standard output is directed to a terminal device, the method for alerting the terminal user is unspecified. When the standard output is not directed to a terminal device, the alert is accomplished by writing the <alert> to standard output (unless the utility description indicates that the use of standard output produces undefined results in this case).	

Clause	Source	Term	Definition	Notes
3. 9	POSIX	Alert Character (<alert>)	A character that in the output stream should cause a terminal to alert its user via a visual or audible notification. It is the character designated by `'\a'` in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the alert function.	
3. 10	POSIX	Alias Name	In the shell command language, a word consisting solely of underscores, digits, and alphabetics from the portable character set and any of the following characters: `'!'`, `'%'`, `','`, `'@'`. Implementations may allow other characters within alias names as an extension.	Note: The Portable Character Set is defined in detail in Portable Character Set.
3. 11	POSIX	Alignment	A requirement that objects of a particular type be located on storage boundaries with addresses that are particular multiples of a byte address.	**Note:** See also the ISO C standard, Section B3.
3. 12	POSIX	Alternate File Access Control Mechanism	An implementation-defined mechanism that is independent of the access control mechanisms defined herein, and which if enabled on a file may either restrict or extend the permissions of a given user. IEEE Std 1003.1-2001 defines when such mechanisms can be enabled and when they are disabled.	Note: File Access Permissions are defined in detail in File Access Permissions.
3. 13	POSIX	Alternate Signal Stack	Memory associated with a thread, established upon request by the implementation for a thread, separate from the thread signal stack, in which signal handlers responding to signals sent to that thread may be executed.	

Clause	Source	Term	Definition	Notes
3. 14	POSIX	Ancillary Data	Protocol-specific, local system-specific, or optional information. The information can be both local or end-to-end significant, header information, part of a data portion, protocol-specific, and implementation or system-specific.	
3. 15	POSIX	Angle Brackets	The characters '<' (left-angle-bracket) and '>' (right-angle-bracket). When used in the phrase "enclosed in angle brackets", the symbol '<' immediately precedes the object to be enclosed, and '>' immediately follows it. When describing these characters in the portable character set, the names <less-than-sign> and <greater-than-sign> are used.	
3. 16	POSIX	Application	A computer program that performs some desired function.	
3. 17	POSIX	Application Address	Endpoint address of a specific application.	
3. 18	POSIX	Application Program Interface (API)	The definition of syntax and semantics for providing computer system services.	
3. 19	POSIX	Appropriate Privileges	An implementation-defined means of associating privileges with a process with regard to the function calls, function call options, and the commands that need special privileges. There may be zero or more such means. These means (or lack thereof) are described in the conformance document.	**Note:** Function calls are defined in the System Interfaces volume of IEEE Std 1003.1-2001, and commands are defined in the Shell and Utilities volume of IEEE Std 1003.1-2001.

Clause	Source	Term	Definition	Notes
3. 20	POSIX	Argument	In the shell command language, a parameter passed to a utility as the equivalent of a single string in the *argv* array created by one of the *exec* functions. An argument is one of the options, option-arguments, or operands following the command name.	**Note:** The Utility Argument Syntax is defined in detail in *Utility Argument Syntax* and the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9.1.1, Command Search and Execution. In the C language, an expression in a function call expression or a sequence of preprocessing tokens in a function-like macro invocation.
3. 21	POSIX	Arm (a Timer)	To start a timer measuring the passage of time, enabling notifying a process when the specified time or time interval has passed.	
3. 22	POSIX	Asterisk	The character `'*'`.	
3. 23	POSIX	Async-Cancel-Safe Function	A function that may be safely invoked by an application while the asynchronous form of cancellation is enabled. No function is async-cancel-safe unless explicitly described as such.	
3. 24	POSIX	Asynchronous Events	Events that occur independently of the execution of the application.	
3. 25	POSIX	Asynchronous Input and Output	A functionality enhancement to allow an application process to queue data input and output commands with asynchronous notification of completion.	
3. 26	POSIX	Async-Signal-Safe Function	A function that may be invoked, without restriction, from signal-catching functions. No function is async-signal-safe unless explicitly described as such.	

Clause	Source	Term	Definition	Notes
3. 27	POSIX	Asynchronously-Generated Signal	A signal that is not attributable to a specific thread. Examples are signals sent via kill(), signals sent from the keyboard, and signals delivered to process groups. Being asynchronous is a property of how the signal was generated and not a property of the signal number. All signals may be generated asynchronously.	Note: The kill() function is defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 28	POSIX	Asynchronous I/O Completion	For an asynchronous read or write operation, when a corresponding synchronous read or write would have completed and when any associated status fields have been updated.	
3. 29	POSIX	Asynchronous I/O Operation	An I/O operation that does not of itself cause the thread requesting the I/O to be blocked from further use of the processor. This implies that the process and the I/O operation may be running concurrently.	
3. 30	POSIX	Authentication	The process of validating a user or process to verify that the user or process is not a counterfeit.	
3. 31	POSIX	Authorization	The process of verifying that a user or process has permission to use a resource in the manner requested. To ensure security, the user or process would also need to be authenticated before granting access.	
3. 32	POSIX	Background Job	See Background Process Group in Background Process Group (or Background Job).	
3. 33	POSIX	Background Process	A process that is a member of a background process group.	
3. 34	POSIX	Background Process Group (or Background Job)	Any process group, other than a foreground process group, that is a member of a session that has established a connection with a controlling terminal.	
3. 35	POSIX	Backquote	The character '`', also known as a grave accent.	

Clause	Source	Term	Definition	Notes
3. 36	POSIX	Backslash	The character '\', also known as a reverse solidus.	
3. 37	POSIX	Backspace Character (<backspace>)	A character that, in the output stream, should cause printing (or displaying) to occur one column position previous to the position about to be printed. If the position about to be printed is at the beginning of the current line, the behavior is unspecified. It is the character designated by '\b' in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the backspace function. The <backspace> defined here is not necessarily the ERASE special character.	Note: Special Characters are defined in detail in Special Characters.
3. 38	POSIX	Barrier	A synchronization object that allows multiple threads to synchronize at a particular point in their execution.	
3. 39	POSIX	Base Character	One of the set of characters defined in the Latin alphabet. In Western European languages other than English, these characters are commonly used with diacritical marks (accents, cedilla, and so on) to extend the range of characters in an alphabet.	
3. 40	POSIX	Basename	The final, or only, filename in a pathname.	
3. 41	POSIX	Basic Regular Expression (BRE)	A regular expression (see Regular Expression) used by the majority of utilities that select strings from a set of character strings.	Note: Basic Regular Expressions are described in detail in Basic Regular Expressions .
3. 42	POSIX	Batch Access List	A list of user IDs and group IDs of those users and groups authorized to place batch jobs in a batch queue. A batch access list is associated with a batch queue. A batch server uses the batch access list of a batch queue as one of the criteria in deciding to put a batch job in a batch queue.	

Clause	Source	Term	Definition	Notes
3. 43	POSIX	Batch Administrator	A user that is authorized to modify all the attributes of queues and jobs and to change the status of a batch server.	
3. 44	POSIX	Batch Client	A computational entity that utilizes batch services by making requests of batch servers. Batch clients often provide the means by which users access batch services, although a batch server may act as a batch client by virtue of making requests of another batch server.	
3. 45	POSIX	Batch Destination	The batch server in a batch system to which a batch job should be sent for processing. Acceptance of a batch job at a batch destination is the responsibility of a receiving batch server. A batch destination may consist of a batch server-specific portion, a network-wide portion, or both. The batch server-specific portion is referred to as the "batch queue". The network-wide portion is referred to as a "batch server name".	
3. 46	POSIX	Batch Destination Identifier	A string that identifies a specific batch destination. A string of characters in the portable character set used to specify a particular batch destination.	Note: The Portable Character Set is defined in detail in Portable Character Set.
3. 47	POSIX	Batch Directive	A line from a file that is interpreted by the batch server. The line is usually in the form of a comment and is an additional means of passing options to the qsub utility.	Note: The qsub utility is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001.
3. 48	POSIX	Batch Job	A set of computational tasks for a computing system. Batch jobs are managed by batch servers. Once created, a batch job may be executing or pending execution. A batch job that is executing has an associated session leader (a process) that initiates and monitors the computational tasks of the batch job.	

Clause	Source	Term	Definition	Notes
3. 49	POSIX	Batch Job Attribute	A named data type whose value affects the processing of a batch job. The values of the attributes of a batch job affect the processing of that job by the batch server that manages the batch job.	
3. 50	POSIX	Batch Job Identifier	A unique name for a batch job. A name that is unique among all other batch job identifiers in a batch system and that identifies the batch server to which the batch job was originally submitted.	
3. 51	POSIX	Batch Job Name	A label that is an attribute of a batch job. The batch job name is not necessarily unique.	
3. 52	POSIX	Batch Job Owner	The username@ hostname of the user submitting the batch job, where username is a user name (see also User Name) and hostname is a network host name.	
3. 53	POSIX	Batch Job Priority	A value specified by the user that may be used by an implementation to determine the order in which batch jobs are selected to be executed. Job priority has a numeric value in the range -1024 to 1023.	**Note:** The batch job priority is not the execution priority (nice value) of the batch job.
3. 54	POSIX	Batch Job State	An attribute of a batch job which determines the types of requests that the batch server that manages the batch job can accept for the batch job. Valid states include QUEUED, RUNNING, HELD, WAITING, EXITING, and TRANSITING.	
3. 55	POSIX	Batch Name Service	A service that assigns batch names that are unique within the batch name space, and that can translate a unique batch name into the location of the named batch entity.	
3. 56	POSIX	Batch Name Space	The environment within which a batch name is known to be unique.	

Clause	Source	Term	Definition	Notes
3. 57	POSIX	Batch Node	A host containing part or all of a batch system. A batch node is a host meeting at least one of the following conditions: (1) Capable of executing a batch client, (2) Contains a routing batch queue, (3) Contains an execution batch queue	
3. 58	POSIX	Batch Operator	A user that is authorized to modify some, but not all, of the attributes of jobs and queues, and may change the status of the batch server.	
3. 59	POSIX	Batch Queue	A manageable object that represents a set of batch jobs and is managed by a single batch server.	Note: A set of batch jobs is called a batch queue largely for historical reasons. Jobs are selected from the batch queue for execution based on attributes such as priority, resource requirements, and hold conditions. See also the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 3.1.2, Batch Queues.
3. 60	POSIX	Batch Queue Attribute	A named data type whose value affects the processing of all batch jobs that are members of the batch queue. A batch queue has attributes that affect the processing of batch jobs that are members of the batch queue.	
3. 61	POSIX	Batch Queue Position	The place, relative to other jobs in the batch queue, occupied by a particular job in a batch queue. This is defined in part by submission time and priority; see also Batch Queue Priority.	
3. 62	POSIX	Batch Queue Priority	The maximum job priority allowed for any batch job in a given batch queue. The batch queue priority is set and may be changed by users with appropriate privilege. The priority is bounded in an implementation-defined manner.	

Clause	Source	Term	Definition	Notes
3. 63	POSIX	Batch Rerunability	An attribute of a batch job indicating that it may be rerun after an abnormal termination from the beginning without affecting the validity of the results.	
3. 64	POSIX	Batch Restart	The action of resuming the processing of a batch job from the point of the last checkpoint. Typically, this is done if the batch job has been interrupted because of a system failure.	
3. 65	POSIX	Batch Server	A computational entity that provides batch services.	
3. 66	POSIX	Batch Server Name	A string of characters in the portable character set used to specify a particular server in a network.	Note: The Portable Character Set is defined in detail in Portable Character Set.
3. 67	POSIX	Batch Service	Computational and organizational services performed by a batch system on behalf of batch jobs. Batch services are of two types: requested and deferred.	**Note:** Batch Services are listed in the Shell and Utilities volume of IEEE Std 1003.1-2001, Table 3-5, Batch Services Summary.
3. 68	POSIX	Batch Service Request	A solicitation of services from a batch client to a batch server. A batch service request may entail the exchange of any number of messages between the batch client and the batch server. When naming specific types of service requests, the term "request" is qualified by the type of request, as in *Queue Batch Job Request* and *Delete Batch Job Request* .	
3. 69	POSIX	Batch Submission	The process by which a batch client requests that a batch server create a batch job via a *Queue Job Request* to perform a specified computational task.	
3. 70	POSIX	Batch System	A collection of one or more batch servers.	

Clause	Source	Term	Definition	Notes
3. 71	POSIX	Batch Target User	The name of a user on the batch destination batch server. The target user is the user name under whose account the batch job is to execute on the destination batch server.	
3. 72	POSIX	Batch User	A user who is authorized to make use of batch services.	
3. 73	POSIX	Bind	The process of assigning a network address to an endpoint.	
3. 74	POSIX	Blank Character (<blank>)	One of the characters that belong to the **blank** character class as defined via the *LC_CTYPE* category in the current locale. In the POSIX locale, a <blank> is either a <tab> or a <space>.	
3. 75	POSIX	Blank Line	A line consisting solely of zero or more <blank>s terminated by a <newline>; see also Empty Line.	
3. 76	POSIX	Blocked Process (or Thread)	A process (or thread) that is waiting for some condition (other than the availability of a processor) to be satisfied before it can continue execution.	
3. 77	POSIX	Blocking	A property of an open file description that causes function calls associated with it to wait for the requested action to be performed before returning.	
3. 78	POSIX	Block-Mode Terminal	A terminal device operating in a mode incapable of the character-at-a-time input and output operations described by some of the standard utilities.	Note: Output Devices and Terminal Types are defined in detail in Output Devices and Terminal Types.
3. 79	POSIX	Block Special File	A file that refers to a device. A block special file is normally distinguished from a character special file by providing access to the device in a manner such that the hardware characteristics of the device are not visible.	

Clause	Source	Term	Definition	Notes
3. 80	POSIX	Braces	The characters '{' (left brace) and '}' (right brace), also known as curly braces. When used in the phrase "enclosed in (curly) braces" the symbol '{' immediately precedes the object to be enclosed, and '}' immediately follows it. When describing these characters in the portable character set, the names <left-brace> and <right-brace> are used.	
3. 81	POSIX	Brackets	The characters '[' (left-bracket) and ']' (right-bracket), also known as square brackets. When used in the phrase "enclosed in (square) brackets" the symbol '[' immediately precedes the object to be enclosed, and ']' immediately follows it. When describing these characters in the portable character set, the names <left-square-bracket> and <right-square-bracket> are used.	
3. 82	POSIX	Broadcast	The transfer of data from one endpoint to several endpoints, as described in RFC 919 and RFC 922.	
3. 83	POSIX	Built-In Utility (or Built-In)	A utility implemented within a shell. The utilities referred to as special built-ins have special qualities. Unless qualified, the term "built-in" includes the special built-in utilities. Regular built-ins are not required to be actually built into the shell on the implementation, but they do have special command-search qualities.	**Note:** Special Built-In Utilities are defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.14, Special Built-In Utilities. Regular Built-In Utilities are defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9.1.1, Command Search and Execution.

Clause	Source	Term	Definition	Notes
3. 84	POSIX	Byte	An individually addressable unit of data storage that is exactly an octet, used to store a character or a portion of a character; see also Character. A byte is composed of a contiguous sequence of 8 bits. The least significant bit is called the "low-order" bit; the most significant is called the "high-order" bit.	**Note:** The definition of byte from the ISO C standard is broader than the above and might accommodate hardware architectures with different sized addressable units than octets.
3. 85	POSIX	Byte Input/Output Functions	The functions that perform byte-oriented input from streams or byte-oriented output to streams: *fgetc* (), *fgets* (), *fprintf* (), *fputc* (), *fputs* (), *fread* (), *fscanf* (), *fwrite* (), *getc* (), *getchar* (), *gets* (), *printf* (), *putc* (), *putchar* (), *puts* (), *scanf* (), *ungetc* (), *vfprintf* (), and *vprintf* ().	**Note:** Functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 86	POSIX	Carriage-Return Character (<carriage-return>)	A character that in the output stream indicates that printing should start at the beginning of the same physical line in which the <carriage-return> occurred. It is the character designated by `'\r'` in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the beginning of the line.	

Clause	Source	Term	Definition	Notes
3. 87	POSIX	Character	A sequence of one or more bytes representing a single graphic symbol or control code.	Note: This term corresponds to the ISO C standard term multi-byte character, where a single-byte character is a special case of a multi-byte character. Unlike the usage in the ISO C standard, character here has no necessary relationship with storage space, and byte is used when storage space is discussed. See the definition of the portable character set in Portable Character Set for a further explanation of the graphical representations of (abstract) characters, as opposed to character encodings.
3. 88	POSIX	Character Array	An array of elements of type **char**.	
3. 89	POSIX	Character Class	A named set of characters sharing an attribute associated with the name of the class. The classes and the characters that they contain are dependent on the value of the *LC_CTYPE* category in the current locale.	Note: The LC_CTYPE category is defined in detail in LC_CTYPE.
3. 90	POSIX	Character Set	A finite set of different characters used for the representation, organization, or control of data.	
3. 91	POSIX	Character Special File	A file that refers to a device. One specific type of character special file is a terminal device file.	Note: The General Terminal Interface is defined in detail in General Terminal Interface .
3. 92	POSIX	Character String	A contiguous sequence of characters terminated by and including the first null byte.	
3. 93	POSIX	Child Process	A new process created (by *fork* (), *posix_spawn* (), *posix_spawnp* (), or *vfork* ()) by a given process. A child process remains the child of the creating process as long as both processes continue to exist.	**Note:** The *fork* (), *posix_spawn* (), *posix_spawnp* (), and *vfork* () functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 94	POSIX	Circumflex	The character '^'.	

Clause	Source	Term	Definition	Notes
3. 95	POSIX	Clock	A software or hardware object that can be used to measure the apparent or actual passage of time. The current value of the time measured by a clock can be queried and, possibly, set to a value within the legal range of the clock.	
3. 96	POSIX	Clock Jump	The difference between two successive distinct values of a clock, as observed from the application via one of the "get time'' operations.	
3. 97	POSIX	Clock Tick	An interval of time; an implementation-defined number of these occur each second. Clock ticks are one of the units that may be used to express a value found in type **clock_t**.	
3. 98	POSIX	Coded Character Set	A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its bit representation.	
3. 99	POSIX	Codeset	The result of applying rules that map a numeric code value to each element of a character set. An element of a character set may be related to more than one numeric code value but the reverse is not true. However, for state-dependent encodings the relationship between numeric code values and elements of a character set may be further controlled by state information. The character set may contain fewer elements than the total number of possible numeric code values; that is, some code values may be unassigned.	Note: Character Encoding is defined in detail in Character Encoding.

Clause	Source	Term	Definition	Notes
3. 100	POSIX	Collating Element	The smallest entity used to determine the logical ordering of character or wide-character strings; see also Collation Sequence. A collating element consists of either a single character, or two or more characters collating as a single entity. The value of the LC_COLLATE category in the current locale determines the current set of collating elements.	
3. 101	POSIX	Collation	The logical ordering of character or wide-character strings according to defined precedence rules. These rules identify a collation sequence between the collating elements, and such additional rules that can be used to order strings consisting of multiple collating elements.	
3. 102	POSIX	Collation Sequence	The relative order of collating elements as determined by the setting of the LC_COLLATE category in the current locale. The collation sequence is used for sorting and is determined from the collating weights assigned to each collating element. In the absence of weights, the collation sequence is the order in which collating elements are specified between order_start and order_end keywords in the LC_COLLATE category. Multi-level sorting is accomplished by assigning elements one or more collation weights, up to the limit {COLL_WEIGHTS_MAX}. On each level, elements may be given the same weight (at the primary level, called an equivalence class; see also Equivalence Class) or be omitted from the sequence. Strings that collate equally using the first assigned weight (primary ordering) are then compared using the next assigned weight (secondary ordering), and so on.	Note: {COLL_WEIGHTS_MAX} is defined in detail in <limits.h>.

Clause	Source	Term	Definition	Notes
3. 103	POSIX	Column Position	A unit of horizontal measure related to characters in a line. It is assumed that each character in a character set has an intrinsic column width independent of any output device. Each printable character in the portable character set has a column width of one. The standard utilities, when used as described in IEEE Std 1003.1-2001, assume that all characters have integral column widths. The column width of a character is not necessarily related to the internal representation of the character (numbers of bits or bytes). The column position of a character in a line is defined as one plus the sum of the column widths of the preceding characters in the line. Column positions are numbered starting from 1.	
3. 104	POSIX	Command	A directive to the shell to perform a particular task.	Note: Shell Commands are defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9, Shell Commands.
3. 105	POSIX	Command Language Interpreter	An interface that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal. It is possible for applications to invoke utilities through a number of interfaces, which are collectively considered to act as command interpreters. The most obvious of these are the *sh* utility and the *system* () function, although *popen* () and the various forms of *exec* may also be considered to behave as interpreters.	**Note:** The *sh* utility is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001. The *system* (), *popen* (), and *exec* functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.

Clause	Source	Term	Definition	Notes
3. 106	POSIX	Composite Graphic Symbol	A graphic symbol consisting of a combination of two or more other graphic symbols in a single character position, such as a diacritical mark and a base character.	
3. 107	POSIX	Condition Variable	A synchronization object which allows a thread to suspend execution, repeatedly, until some associated predicate becomes true. A thread whose execution is suspended on a condition variable is said to be blocked on the condition variable.	
3. 108	POSIX	Connection	An association established between two or more endpoints for the transfer of data	
3. 109	POSIX	Connection Mode	The transfer of data in the context of a connection; see also Connectionless Mode.	
3. 110	POSIX	Connectionless Mode	The transfer of data other than in the context of a connection; see also Connection Mode and Datagram.	
3. 111	POSIX	Control Character	A character, other than a graphic character, that affects the recording, processing, transmission, or interpretation of text.	
3. 112	POSIX	Control Operator	In the shell command language, a token that performs a control function. It is one of the following symbols: `&&` `(` `)` `;` `;;` `newline` `|` `||` The end-of-input indicator used internally by the shell is also considered a control operator.	Note: Token Recognition is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.3, Token Recognition.
3. 113	POSIX	Controlling Process	The session leader that established the connection to the controlling terminal. If the terminal subsequently ceases to be a controlling terminal for this session, the session leader ceases to be the controlling process.	

Clause	Source	Term	Definition	Notes
3. 114	POSIX	Controlling Terminal	A terminal that is associated with a session. Each session may have at most one controlling terminal associated with it, and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal cause signals to be sent to all processes in the process group associated with the controlling terminal.	Note: The General Terminal Interface is defined in detail in General Terminal Interface .
3. 115	POSIX	Conversion Descriptor	A per-process unique value used to identify an open codeset conversion.	
3. 116	POSIX	Core File	A file of unspecified format that may be generated when a process terminates abnormally.	
3. 117	POSIX	CPU Time (Execution Time)	The time spent executing a process or thread, including the time spent executing system services on behalf of that process or thread. If the Threads option is supported, then the value of the CPU-time clock for a process is implementation-defined. With this definition the sum of all the execution times of all the threads in a process might not equal the process execution time, even in a single-threaded process, because implementations may differ in how they account for time during context switches or for other reasons.	
3. 118	POSIX	CPU-Time Clock	A clock that measures the execution time of a particular process or thread.	
3. 119	POSIX	CPU-Time Timer	A timer attached to a CPU-time clock.	
3. 120	POSIX	Current Job	In the context of job control, the job that will be used as the default for the *fg* or *bg* utilities. There is at most one current job; see also Job Control Job ID.	

Clause	Source	Term	Definition	Notes
3. 121	POSIX	Current Working Directory	See Working Directory in Working Directory (or Current Working Directory).	
3. 122	POSIX	Cursor Position	The line and column position on the screen denoted by the terminal's cursor.	
3. 123	POSIX	Datagram	A unit of data transferred from one endpoint to another in connectionless mode service.	
3. 124	POSIX	Data Segment	Memory associated with a process, that can contain dynamically allocated data.	
3. 125	POSIX	Deferred Batch Service	A service that is performed as a result of events that are asynchronous with respect to requests.	**Note:** Once a batch job has been created, it is subject to deferred services.
3. 126	POSIX	Device	A computer peripheral or an object that appears to the application as such.	
3. 127	POSIX	Device ID	A non-negative integer used to identify a device.	
3. 128	POSIX	Directory	A file that contains directory entries. No two directory entries in the same directory have the same name.	
3. 129	POSIX	Directory Entry (or Link)	An object that associates a filename with a file. Several directory entries can associate names with the same file.	
3. 130	POSIX	Directory Stream	A sequence of all the directory entries in a particular directory. An open directory stream may be implemented using a file descriptor.	
3. 131	POSIX	Disarm (a Timer)	To stop a timer from measuring the passage of time, disabling any future process notifications (until the timer is armed again).	
3. 132	POSIX	Display	To output to the user's terminal. If the output is not directed to a terminal, the results are undefined.	
3. 133	POSIX	Display Line	A line of text on a physical device or an emulation thereof. Such a line will have a maximum number of characters which can be presented.	**Note:** This may also be written as "line on the display".

Clause	Source	Term	Definition	Notes
3. 134	POSIX	Dollar Sign	The character `'$'`.	
3. 135	POSIX	Dot	In the context of naming files, the filename consisting of a single dot character (`'.'`).	**Note:** In the context of shell special built-in utilities, see *dot* in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.14, Special Built-In Utilities. Pathname Resolution is defined in detail in *Pathname Resolution* .
3. 136	POSIX	Dot-Dot	The filename consisting solely of two dot characters (`".."`).	Note: Pathname Resolution is defined in detail in Pathname Resolution.
3. 137	POSIX	Double-Quote	The character `'"'`, also known as quotation-mark.	**Note:** The "double" adjective in this term refers to the two strokes in the character glyph. IEEE Std 1003.1-2001 never uses the term "double-quote" to refer to two apostrophes or quotation marks.
3. 138	POSIX	Downshifting	The conversion of an uppercase character that has a single-character lowercase representation into this lowercase representation.	
3. 139	POSIX	Driver	A module that controls data transferred to and received from devices.	**Note:** Drivers are traditionally written to be a part of the system implementation, although they are frequently written separately from the writing of the implementation. A driver may contain processor-specific code, and therefore be non-portable.
3. 140	POSIX	Effective Group ID	An attribute of a process that is used in determining various permissions, including file access permissions; see also Group ID.	
3. 141	POSIX	Effective User ID	An attribute of a process that is used in determining various permissions, including file access permissions; see also User ID.	

Clause	Source	Term	Definition	Notes
3. 142	POSIX	Eight-Bit Transparency	The ability of a software component to process 8-bit characters without modifying or utilizing any part of the character in a way that is inconsistent with the rules of the current coded character set.	
3. 143	POSIX	Empty Directory	A directory that contains, at most, directory entries for dot and dot-dot, and has exactly one link to it, in dot-dot. No other links to the directory may exist. It is unspecified whether an implementation can ever consider the root directory to be empty.	
3. 144	POSIX	Empty Line	A line consisting of only a <newline>; see also Blank Line.	
3. 145	POSIX	Empty String (or Null String)	A string whose first byte is a null byte.	
3. 146	POSIX	Empty Wide-Character String	A wide-character string whose first element is a null wide-character code.	
3. 147	POSIX	Encoding Rule	The rules used to convert between wide-character codes and multi-byte character codes.	Note: Stream Orientation and Encoding Rules are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.5.2, Stream Orientation and Encoding Rules.
3. 148	POSIX	Entire Regular Expression	The concatenated set of one or more basic regular expressions or extended regular expressions that make up the pattern specified for string selection.	Note: Regular Expressions are defined in detail in Regular Expressions.
3. 149	POSIX	Epoch	The time zero hours, zero minutes, zero seconds, on January 1, 1970 Coordinated Universal Time (UTC).	Note: See also Seconds Since the Epoch defined in Seconds Since the Epoch.

Clause	Source	Term	Definition	Notes
3. 150	POSIX	Equivalence Class	A set of collating elements with the same primary collation weight. Elements in an equivalence class are typically elements that naturally group together, such as all accented letters based on the same base letter. The collation order of elements within an equivalence class is determined by the weights assigned on any subsequent levels after the primary weight.	
3. 151	POSIX	Era	A locale-specific method for counting and displaying years.	Note: The LC_TIME category is defined in detail in LC_TIME.
3. 152	POSIX	Event Management	The mechanism that enables applications to register for and be made aware of external events such as data becoming available for reading.	
3. 153	POSIX	Executable File	A regular file acceptable as a new process image file by the equivalent of the *exec* family of functions, and thus usable as one form of a utility. The standard utilities described as compilers can produce executable files, but other unspecified methods of producing executable files may also be provided. The internal format of an executable file is unspecified, but a conforming application cannot assume an executable file is a text file.	
3. 154	POSIX	Execute	To perform command search and execution actions, as defined in the Shell and Utilities volume of IEEE Std 1003.1-2001; see also Invoke.	Note: Command Search and Execution is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9.1.1, Command Search and Execution.
3. 155	POSIX	Execution Time	See CPU Time in CPU Time (Execution Time).	
3. 156	POSIX	Execution Time Monitoring	A set of execution time monitoring primitives that allow online measuring of thread and process execution times.	

Clause	Source	Term	Definition	Notes
3. 157	POSIX	Expand	In the shell command language, when not qualified, the act of applying word expansions.	Note: Word Expansions are defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6, Word Expansions.
3. 158	POSIX	Extended Regular Expression (ERE)	A regular expression (see also Regular Expression) that is an alternative to the Basic Regular Expression using a more extensive syntax, occasionally used by some utilities.	Note: Extended Regular Expressions are described in detail in Extended Regular Expressions.
3. 159	POSIX	Extended Security Controls	Implementation-defined security controls allowed by the file access permission and appropriate privilege (see also Appropriate Privileges) mechanisms, through which an implementation can support different security policies from those described in IEEE Std 1003.1-2001.	**Note:** See also Extended Security Controls defined in *Extended Security Controls* . File Access Permissions are defined in detail in *File Access Permissions* .
3. 160	POSIX	Feature Test Macro	A macro used to determine whether a particular set of features is included from a header.	Note: See also the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.2, The Compilation Environment.

Clause	Source	Term	Definition	Notes
3. 161	POSIX	Field	In the shell command language, a unit of text that is the result of parameter expansion, arithmetic expansion, command substitution, or field splitting. During command processing, the resulting fields are used as the command name and its arguments.	**Note:** Parameter Expansion is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6.2, Parameter Expansion. Arithmetic Expansion is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6.4, Arithmetic Expansion. Command Substitution is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6.3, Command Substitution. Field Splitting is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6.5, Field Splitting. For further information on command processing, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9.1, Simple Commands.
3. 162	POSIX	FIFO Special File (or FIFO)	A type of file with the property that data written to such a file is read on a first-in-first-out basis.	**Note:** Other characteristics of FIFOs are described in the System Interfaces volume of IEEE Std 1003.1-2001, *lseek* (), *open* (), *read* (), and *write* ().
3. 163	POSIX	File	An object that can be written to, or read from, or both. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, symbolic link, socket, and directory. Other types of files may be supported by the implementation.	
3. 164	POSIX	File Description	See Open File Description in Open File Description.	

Clause	Source	Term	Definition	Notes
3. 165	POSIX	File Descriptor	A per-process unique, non-negative integer used to identify an open file for the purpose of file access. The value of a file descriptor is from zero to {OPEN_MAX}. A process can have no more than {OPEN_MAX} file descriptors open simultaneously. File descriptors may also be used to implement message catalog descriptors and directory streams; see also Open File Description.	Note: {OPEN_MAX} is defined in detail in <limits.h>.
3. 166	POSIX	File Group Class	The property of a file indicating access permissions for a process related to the group identification of a process. A process is in the file group class of a file if the process is not in the file owner class and if the effective group ID or one of the supplementary group IDs of the process matches the group ID associated with the file. Other members of the class may be implementation-defined.	
3. 167	POSIX	File Mode	An object containing the file mode bits and file type of a file.	Note: File mode bits and file types are defined in detail in <sys/stat.h>.
3. 168	POSIX	File Mode Bits	A file's file permission bits: set-user-ID-on-execution bit (S_ISUID), set-group-ID-on-execution bit (S_ISGID), and, on directories, the restricted deletion flag bit (S_ISVTX).	Note: File Mode Bits are defined in detail in <sys/stat.h>.
3. 169	POSIX	Filename	A name consisting of 1 to {NAME_MAX} bytes used to name a file. The characters composing the name may be selected from the set of all character values excluding the slash character and the null byte. The filenames dot and dot-dot have special meaning. A filename is sometimes referred to as a "pathname component".	Note: Pathname Resolution is defined in detail in Pathname Resolution.

Clause	Source	Term	Definition	Notes
3. 170	POSIX	Filename Portability	Filenames should be constructed from the portable filename character set because the use of other characters can be confusing or ambiguous in certain contexts. (For example, the use of a colon (':') in a pathname could cause ambiguity if that pathname were included in a *PATH* definition.)	
3. 171	POSIX	File Offset	The byte position in the file where the next I/O operation begins. Each open file description associated with a regular file, block special file, or directory has a file offset. A character special file that does not refer to a terminal device may have a file offset. There is no file offset specified for a pipe or FIFO.	
3. 172	POSIX	File Other Class	The property of a file indicating access permissions for a process related to the user and group identification of a process. A process is in the file other class of a file if the process is not in the file owner class or file group class.	
3. 173	POSIX	File Owner Class	The property of a file indicating access permissions for a process related to the user identification of a process. A process is in the file owner class of a file if the effective user ID of the process matches the user ID of the file.	
3. 174	POSIX	File Permission Bits	Information about a file that is used, along with other information, to determine whether a process has read, write, or execute/search permission to a file. The bits are divided into three parts: owner, group, and other. Each part is used with the corresponding file class of processes. These bits are contained in the file mode.	**Note:** File modes are defined in detail in <*sys/stat.h*> . File Access Permissions are defined in detail in *File Access Permissions* .

Clause	Source	Term	Definition	Notes
3. 175	POSIX	File Serial Number	A per-file system unique identifier for a file.	
3. 176	POSIX	File System	A collection of files and certain of their attributes. It provides a name space for file serial numbers referring to those files.	
3. 177	POSIX	File Type	See File in File.	
3. 178	POSIX	Filter	A command whose operation consists of reading data from standard input or a list of input files and writing data to standard output. Typically, its function is to perform some transformation on the data stream.	
3. 179	POSIX	First Open (of a File)	When a process opens a file that is not currently an open file within any process.	
3. 180	POSIX	Flow Control	The mechanism employed by a communications provider that constrains a sending entity to wait until the receiving entities can safely receive additional data without loss.	
3. 181	POSIX	Foreground Job	See Foreground Process Group in Foreground Process Group (or Foreground Job).	
3. 182	POSIX	Foreground Process	A process that is a member of a foreground process group.	
3. 183	POSIX	Foreground Process Group (or Foreground Job)	A process group whose member processes have certain privileges, denied to processes in background process groups, when accessing their controlling terminal. Each session that has established a connection with a controlling terminal has at most one process group of the session as the foreground process group of that controlling terminal.	Note: The General Terminal Interface is defined in detail in General Terminal Interface .
3. 184	POSIX	Foreground Process Group ID	The process group ID of the foreground process group.	

Clause	Source	Term	Definition	Notes
3. 185	POSIX	Form-Feed Character (<form-feed>)	A character that in the output stream indicates that printing should start on the next page of an output device. It is the character designated by `'\f'` in the C language. If the <form-feed> is not the first character of an output line, the result is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the next page.	
3. 186	POSIX	Graphic Character	A member of the **graph** character class of the current locale.	Note: The graph character class is defined in detail in LC_CTYPE.
3. 187	POSIX	Group Database	A system database that contains at least the following information for each group ID: (1) Group name, (2) Numerical group ID, (3) List of users allowed in the group. The list of users allowed in the group is used by the newgrp utility.	Note: The newgrp utility is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001.
3. 188	POSIX	Group ID	A non-negative integer, which can be contained in an object of type **gid_t**, that is used to identify a group of system users. Each system user is a member of at least one group. When the identity of a group is associated with a process, a group ID value is referred to as a real group ID, an effective group ID, one of the supplementary group IDs, or a saved set-group-ID.	
3. 189	POSIX	Group Name	A string that is used to identify a group; see also Group Database. To be portable across conforming systems, the value is composed of characters from the portable filename character set. The hyphen should not be used as the first character of a portable group name.	

Clause	Source	Term	Definition	Notes
3. 190	POSIX	Hard Limit	A system resource limitation that may be reset to a lesser or greater limit by a privileged process. A non-privileged process is restricted to only lowering its hard limit.	
3. 191	POSIX	Hard Link	The relationship between two directory entries that represent the same file; see also Directory Entry (or Link). The result of an execution of the *ln* utility (without the **-s** option) or the *link* () function. This term is contrasted against symbolic link; see also Symbolic Link.	
3. 192	POSIX	Home Directory	The directory specified by the *HOME* environment variable.	
3. 193	POSIX	Host Byte Order	The arrangement of bytes in any integer type when using a specific machine architecture.	Note: Two common methods of byte ordering are big-endian and little-endian. Big-endian is a format for storage of binary data in which the most significant byte is placed first, with the rest in descending order. Little-endian is a format for storage or transmission of binary data in which the least significant byte is placed first, with the rest in ascending order. See also Host and Network Byte Orders.
3. 194	POSIX	Incomplete Line	A sequence of one or more non- <newline>s at the end of the file.	
3. 195	POSIX	Inf	A value representing +infinity or a value representing -infinity that can be stored in a floating type. Not all systems support the Inf values.	

Clause	Source	Term	Definition	Notes
3. 196	POSIX	Instrumented Application	An application that contains at least one call to the trace point function posix_trace_event(). Each process of an instrumented application has a mapping of trace event names to trace event type identifiers. This mapping is used by the trace stream that is created for that process.	
3. 197	POSIX	Interactive Shell	A processing mode of the shell that is suitable for direct user interaction.	
3. 198	POSIX	Internationalization	The provision within a computer program of the capability of making itself adaptable to the requirements of different native languages, local customs, and coded character sets.	
3. 199	POSIX	Interprocess Communication	A functionality enhancement to add a high-performance, deterministic interprocess communication facility for local communication.	
3. 200	POSIX	Invoke	To perform command search and execution actions, except that searching for shell functions and special built-in utilities is suppressed; see also Execute.	Note: Command Search and Execution is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9.1.1, Command Search and Execution.
3. 201	POSIX	Job	A set of processes, comprising a shell pipeline, and any processes descended from it, that are all in the same process group.	Note: See also the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9.2, Pipelines.
3. 202	POSIX	Job Control	A facility that allows users selectively to stop (suspend) the execution of processes and continue (resume) their execution at a later point. The user typically employs this facility via the interactive interface jointly supplied by the terminal I/O driver and a command interpreter.	
3. 203	POSIX	Job Control Job ID	A handle that is used to refer to a job. The job control job ID can be any of the forms shown in table Posix table 203.	

Clause	Source	Term	Definition	Notes
3. 204	POSIX	Last Close (of a File)	When a process closes a file, resulting in the file not being an open file within any process.	
3. 205	POSIX	Line	A sequence of zero or more non- <newline>s plus a terminating <newline>.	
3. 206	POSIX	Linger	A period of time before terminating a connection, to allow outstanding data to be transferred.	
3. 207	POSIX	Link	See Directory Entry in Directory Entry (or Link).	
3. 208	POSIX	Link Count	The number of directory entries that refer to a particular file.	
3. 209	POSIX	Local Customs	The conventions of a geographical area or territory for such things as date, time, and currency formats.	
3. 210	POSIX	Local Interprocess Communication (Local IPC)	The transfer of data between processes in the same system.	
3. 211	POSIX	Locale	The definition of the subset of a user's environment that depends on language and cultural conventions.	Note: Locales are defined in detail in Locale.
3. 212	POSIX	Localization	The process of establishing information within a computer system specific to the operation of particular native languages, local customs, and coded character sets.	
3. 213	POSIX	Login	The unspecified activity by which a user gains access to the system. Each login is associated with exactly one login name.	
3. 214	POSIX	Login Name	A user name that is associated with a login.	

Clause	Source	Term	Definition	Notes
3. 215	POSIX	Map	To create an association between a page-aligned range of the address space of a process and some memory object, such that a reference to an address in that range of the address space results in a reference to the associated memory object. The mapped memory object is not necessarily memory-resident.	
3. 216	POSIX	Marked Message	A STREAMs message on which a certain flag is set. Marking a message gives the application protocol-specific information. An application can use ioctl() to determine whether a given message is marked.	Note: The ioctl() function is defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 217	POSIX	Matched	A state applying to a sequence of zero or more characters when the characters in the sequence correspond to a sequence of characters defined by a basic regular expression or extended regular expression pattern.	Note: Regular Expressions are defined in detail in Regular Expressions.
3. 218	POSIX	Memory Mapped Files	A facility to allow applications to access files as part of the address space.	
3. 219	POSIX	Memory Object	One of: (1) A file (see File), (2) A shared memory object (see Shared Memory Object), (3) A typed memory object (see Typed Memory Object). When used in conjunction with *mmap* (), a memory object appears in the address space of the calling process.	Note: The mmap() function is defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 220	POSIX	Memory-Resident	The process of managing the implementation in such a way as to provide an upper bound on memory access times.	
3. 221	POSIX	Message	In the context of programmatic message passing, information that can be transferred between processes or threads by being added to and removed from a message queue. A message consists of a fixed-size message buffer.	

Clause	Source	Term	Definition	Notes
3. 222	POSIX	Message Catalog	In the context of providing natural language messages to the user, a file or storage area containing program messages, command prompts, and responses to prompts for a particular native language, territory, and codeset.	
3. 223	POSIX	Message Catalog Descriptor	In the context of providing natural language messages to the user, a per-process unique value used to identify an open message catalog. A message catalog descriptor may be implemented using a file descriptor.	
3. 224	POSIX	Message Queue	In the context of programmatic message passing, an object to which messages can be added and removed. Messages may be removed in the order in which they were added or in priority order.	
3. 225	POSIX	Mode	A collection of attributes that specifies a file's type and its access permissions.	Note: File Access Permissions are defined in detail in File Access Permissions.
3. 226	POSIX	Monotonic Clock	A clock whose value cannot be set via clock_settime() and which cannot have negative clock jumps.	
3. 227	POSIX	Mount Point	Either the system root directory or a directory for which the *st_dev* field of structure **stat** differs from that of its parent directory.	Note: The stat structure is defined in detail in <sys/stat.h>.
3. 228	POSIX	Multi-Character Collating Element	A sequence of two or more characters that collate as an entity. For example, in some coded character sets, an accented character is represented by a non-spacing accent, followed by the letter. Other examples are the Spanish elements *ch* and *ll*.	

Clause	Source	Term	Definition	Notes
3. 229	POSIX	Mutex	A synchronization object used to allow multiple threads to serialize their access to shared data. The name derives from the capability it provides; namely, mutual-exclusion. The thread that has locked a mutex becomes its owner and remains the owner until that same thread unlocks the mutex.	
3. 230	POSIX	Name	In the shell command language, a word consisting solely of underscores, digits, and alphabetics from the portable character set. The first character of a name is not a digit.	Note: The Portable Character Set is defined in detail in Portable Character Set.
3. 231	POSIX	Named STREAM	A STREAMS-based file descriptor that is attached to a name in the file system name space. All subsequent operations on the named STREAM act on the STREAM that was associated with the file descriptor until the name is disassociated from the STREAM.	
3. 232	POSIX	NaN (Not a Number)	A set of values that may be stored in a floating type but that are neither Inf nor valid floating-point numbers. Not all systems support NaN values.	
3. 233	POSIX	Native Language	A computer user's spoken or written language, such as American English, British English, Danish, Dutch, French, German, Italian, Japanese, Norwegian, or Swedish.	
3. 234	POSIX	Negative Response	An input string that matches one of the responses acceptable to the *LC_MESSAGES* category keyword **noexpr**, matching an extended regular expression in the current locale.	Note: The LC_MESSAGES category is defined in detail in LC_MESSAGES.

Clause	Source	Term	Definition	Notes
3. 235	POSIX	Network	A collection of interconnected hosts.	**Note:** The term "network" in IEEE Std 1003.1-2001 is used to refer to the network of hosts. The term "batch system" is used to refer to the network of batch servers.
3. 236	POSIX	Network Address	A network-visible identifier used to designate specific endpoints in a network. Specific endpoints on host systems have addresses, and host systems may also have addresses.	
3. 237	POSIX	Network Byte Order	The way of representing any integer type such that, when transmitted over a network via a network endpoint, the **int** type is transmitted as an appropriate number of octets with the most significant octet first, followed by any other octets in descending order of significance.	Note: This order is more commonly known as big-endian ordering. See also Host and Network Byte Orders.
3. 238	POSIX	Newline Character (<newline>)	A character that in the output stream indicates that printing should start at the beginning of the next line. It is the character designated by `'\n'` in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the next line.	

Clause	Source	Term	Definition	Notes
3. 239	POSIX	Nice Value	A number used as advice to the system to alter process scheduling. Numerically smaller values give a process additional preference when scheduling a process to run. Numerically larger values reduce the preference and make a process less likely to run. Typically, a process with a smaller nice value runs to completion more quickly than an equivalent process with a higher nice value. The symbol {NZERO} specifies the default nice value of the system.	
3. 240	POSIX	Non-Blocking	A property of an open file description that causes function calls involving it to return without delay when it is detected that the requested action associated with the function call cannot be completed without unknown delay.	Note: The exact semantics are dependent on the type of file associated with the open file description. For data reads from devices such as ttys and FIFOs, this property causes the read to return immediately when no data was available. Similarly, for writes, it causes the call to return immediately when the thread would otherwise be delayed in the write operation; for example, because no space was available. For networking, it causes functions not to await protocol events (for example, acknowledgements) to occur. See also the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.10.7, Socket I/O Mode.
3. 241	POSIX	Non-Spacing Characters	A character, such as a character representing a diacritical mark in the ISO/IEC 6937:2001 standard coded character set, which is used in combination with other characters to form composite graphic symbols.	
3. 242	POSIX	NUL	A character with all bits set to zero.	
3. 243	POSIX	Null Byte	A byte with all bits set to zero.	

Clause	Source	Term	Definition	Notes
3. 244	POSIX	Null Pointer	The value that is obtained by converting the number 0 into a pointer; for example, (**void** *) 0. The C language guarantees that this value does not match that of any legitimate pointer, so it is used by many functions that return pointers to indicate an error.	
3. 245	POSIX	Null String	See Empty String in Empty String (or Null String).	
3. 246	POSIX	Null Wide-Character Code	A wide-character code with all bits set to zero.	
3. 247	POSIX	Number Sign	The character '#', also known as hash sign.	
3. 248	POSIX	Object File	A regular file containing the output of a compiler, formatted as input to a linkage editor for linking with other object files into an executable form. The methods of linking are unspecified and may involve the dynamic linking of objects at runtime. The internal format of an object file is unspecified, but a conforming application cannot assume an object file is a text file.	
3. 249	POSIX	Octet	Unit of data representation that consists of eight contiguous bits.	
3. 250	POSIX	Offset Maximum	An attribute of an open file description representing the largest value that can be used as a file offset.	
3. 251	POSIX	Opaque Address	An address such that the entity making use of it requires no details about its contents or format.	
3. 252	POSIX	Open File	A file that is currently associated with a file descriptor.	

Clause	Source	Term	Definition	Notes
3. 253	POSIX	Open File Description	A record of how a process or group of processes is accessing a file. Each file descriptor refers to exactly one open file description, but an open file description can be referred to by more than one file descriptor. The file offset, file status, and file access modes are attributes of an open file description.	
3. 254	POSIX	Operand	An argument to a command that is generally used as an object supplying information to a utility necessary to complete its processing. Operands generally follow the options in a command line.	Note: Utility Argument Syntax is defined in detail in Utility Argument Syntax.
3. 255	POSIX	Operator	In the shell command language, either a control operator or a redirection operator.	
3. 256	POSIX	Option	An argument to a command that is generally used to specify changes in the utility's default behavior.	Note: Utility Argument Syntax is defined in detail in Utility Argument Syntax.
3. 257	POSIX	Option-Argument	A parameter that follows certain options. In some cases an option-argument is included within the same argument string as the option-in most cases it is the next argument.	Note: Utility Argument Syntax is defined in detail in Utility Argument Syntax.
3. 258	POSIX	Orientation	A stream has one of three orientations: unoriented, byte-oriented, or wide-oriented.	Note: For further information, see the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.5.2, Stream Orientation and Encoding Rules.
3. 259	POSIX	Orphaned Process Group	A process group in which the parent of every member is either itself a member of the group or is not a member of the group's session.	

Clause	Source	Term	Definition	Notes
3. 260	POSIX	Page	The granularity of process memory mapping or locking. Physical memory and memory objects can be mapped into the address space of a process on page boundaries and in integral multiples of pages. Process address space can be locked into memory (made memory-resident) on page boundaries and in integral multiples of pages.	
3. 261	POSIX	Page Size	The size, in bytes, of the system unit of memory allocation, protection, and mapping. On systems that have segment rather than page-based memory architectures, the term "page" means a segment.	
3. 262	POSIX	Parameter	In the shell command language, an entity that stores values. There are three types of parameters: variables (named parameters), positional parameters, and special parameters. Parameter expansion is accomplished by introducing a parameter with the '$' character.	Note: See also the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.5, Parameters and Variables. In the C language, an object declared as part of a function declaration or definition that acquires a value on entry to the function, or an identifier following the macro name in a function-like macro definition.
3. 263	POSIX	Parent Directory	When discussing a given directory, the directory that both contains a directory entry for the given directory and is represented by the pathname dot-dot in the given directory. When discussing other types of files, a directory containing a directory entry for the file under discussion. This concept does not apply to dot and dot-dot.	
3. 264	POSIX	Parent Process	The process which created (or inherited) the process under discussion.	

Clause	Source	Term	Definition	Notes
3. 265	POSIX	Parent Process ID	An attribute of a new process identifying the parent of the process. The parent process ID of a process is the process ID of its creator, for the lifetime of the creator. After the creator's lifetime has ended, the parent process ID is the process ID of an implementation-defined system process.	
3. 266	POSIX	Pathname	A character string that is used to identify a file. In the context of IEEE Std 1003.1-2001, a pathname consists of, at most, {PATH_MAX} bytes, including the terminating null byte. It has an optional beginning slash, followed by zero or more filenames separated by slashes. A pathname may optionally contain one or more trailing slashes. Multiple successive slashes are considered to be the same as one slash.	Note: Pathname Resolution is defined in detail in Pathname Resolution.
3. 267	POSIX	Pathname Component	See Filename in Filename.	
3. 268	POSIX	Path Prefix	A pathname, with an optional ending slash, that refers to a directory.	
3. 269	POSIX	Pattern	A sequence of characters used either with regular expression notation or for pathname expansion, as a means of selecting various character strings or pathnames, respectively.	**Note:** Regular Expressions are defined in detail in *Regular Expressions* . See also the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6.6, Pathname Expansion. The syntaxes of the two types of patterns are similar, but not identical; IEEE Std 1003.1-2001 always indicates the type of pattern being referred to in the immediate context of the use of the term.

Clause	Source	Term	Definition	Notes
3. 270	POSIX	Period	The character '.'. The term "period" is contrasted with dot (see also Dot), which is used to describe a specific directory entry.	
3. 271	POSIX	Permissions	Attributes of an object that determine the privilege necessary to access or manipulate the object.	Note: File Access Permissions are defined in detail in File Access Permissions.
3. 272	POSIX	Persistence	A mode for semaphores, shared memory, and message queues requiring that the object and its state (including data, if any) are preserved after the object is no longer referenced by any process. Persistence of an object does not imply that the state of the object is maintained across a system crash or a system reboot.	
3. 273	POSIX	Pipe	An object accessed by one of the pair of file descriptors created by the pipe() function. Once created, the file descriptors can be used to manipulate it, and it behaves identically to a FIFO special file when accessed in this way. It has no name in the file hierarchy.	Note: The pipe() function is defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 274	POSIX	Polling	A scheduling scheme whereby the local process periodically checks until the pre-specified events (for example, read, write) have occurred.	
3. 275	POSIX	Portable Character Set	The collection of characters that are required to be present in all locales supported by conforming systems.	**Note:** The Portable Character Set is defined in detail in *Portable Character Set* . This term is contrasted against the smaller portable filename character set; see also Portable Filename Character Set.

Clause	Source	Term	Definition	Notes
3. 276	POSIX	Portable Filename Character Set	The set of characters from which portable filenames are constructed. {A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 . _ -} The last three characters are the period, underscore, and hyphen characters, respectively.	
3. 277	POSIX	Positional Parameter	In the shell command language, a parameter denoted by a single digit or one or more digits in curly braces.	Note: For further information, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.5.1, Positional Parameters.
3. 278	POSIX	Preallocation	The reservation of resources in a system for a particular use. Preallocation does not imply that the resources are immediately allocated to that use, but merely indicates that they are guaranteed to be available in bounded time when needed.	
3. 279	POSIX	Preempted Process (or Thread)	A running thread whose execution is suspended due to another thread becoming runnable at a higher priority.	
3. 280	POSIX	Previous Job	In the context of job control, the job that will be used as the default for the *fg* or *bg* utilities if the current job exits. There is at most one previous job; see also Job Control Job ID.	
3. 281	POSIX	Printable Character	One of the characters included in the **print** character classification of the *LC_CTYPE* category in the current locale.	Note: The LC_CTYPE category is defined in detail in LC_CTYPE.
3. 282	POSIX	Printable File	A text file consisting only of the characters included in the **print** and **space** character classifications of the *LC_CTYPE* category and the <backspace>, all in the current locale.	Note: The LC_CTYPE category is defined in detail in LC_CTYPE.

Clause	Source	Term	Definition	Notes
3. 283	POSIX	Priority	A non-negative integer associated with processes or threads whose value is constrained to a range defined by the applicable scheduling policy. Numerically higher values represent higher priorities.	
3. 284	POSIX	Priority Band	The queuing order applied to normal priority STREAMS messages. High priority STREAMS messages are not grouped by priority bands. The only differentiation made by the STREAMS mechanism is between zero and non-zero bands, but specific protocol modules may differentiate between priority bands.	
3. 285	POSIX	Priority Inversion	A condition in which a thread that is not voluntarily suspended (waiting for an event or time delay) is not running while a lower priority thread is running. Such blocking of the higher priority thread is often caused by contention for a shared resource.	
3. 286	POSIX	Priority Scheduling	A performance and determinism improvement facility to allow applications to determine the order in which threads that are ready to run are granted access to processor resources.	
3. 287	POSIX	Priority-Based Scheduling	Scheduling in which the selection of a running thread is determined by the priorities of the runnable processes or threads.	
3. 288	POSIX	Privilege	See Appropriate Privileges in Appropriate Privileges.	

Clause	Source	Term	Definition	Notes
3. 289	POSIX	Process	An address space with one or more threads executing within that address space, and the required system resources for those threads.	**Note:** Many of the system resources defined by IEEE Std 1003.1-2001 are shared among all of the threads within a process. These include the process ID, the parent process ID, process group ID, session membership, real, effective, and saved set-user-ID, real, effective, and saved set-group-ID, supplementary group IDs, current working directory, root directory, file mode creation mask, and file descriptors.
3. 290	POSIX	Process Group	A collection of processes that permits the signaling of related processes. Each process in the system is a member of a process group that is identified by a process group ID. A newly created process joins the process group of its creator.	
3. 291	POSIX	Process Group ID	The unique positive integer identifier representing a process group during its lifetime.	Note: See also Process Group ID Reuse defined in Process ID Reuse.
3. 292	POSIX	Process Group Leader	A process whose process ID is the same as its process group ID.	
3. 293	POSIX	Process Group Lifetime	A period of time that begins when a process group is created and ends when the last remaining process in the group leaves the group, due either to the end of the last process' lifetime or to the last remaining process calling the *setsid* () or *setpgid* () functions.	**Note:** The *setsid* () and *setpgid* () functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 294	POSIX	Process ID	The unique positive integer identifier representing a process during its lifetime.	Note: See also Process ID Reuse defined in Process ID Reuse.

Clause	Source	Term	Definition	Notes
3. 295	POSIX	Process Lifetime	The period of time that begins when a process is created and ends when its process ID is returned to the system. After a process is created by *fork* (), *posix_spawn* (), *posix_spawnp* (), or *vfork* (), it is considered active. At least one thread of control and address space exist until it terminates. It then enters an inactive state where certain resources may be returned to the system, although some resources, such as the process ID, are still in use. When another process executes a *wait* (), *waitid* (), or *waitpid* () function for an inactive process, the remaining resources are returned to the system. The last resource to be returned to the system is the process ID. At this time, the lifetime of the process ends.	**Note:** The *fork* (), *posix_spawn* (), *posix_spawnp* (), *vfork* (), *wait* (), *waitid* (), and *waitpid* () functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 296	POSIX	Process Memory Locking	A performance improvement facility to bind application programs into the high-performance random access memory of a computer system. This avoids potential latencies introduced by the operating system in storing parts of a program that were not recently referenced on secondary memory devices.	
3. 297	POSIX	Process Termination	There are two kinds of process termination: (1) Normal termination occurs by a return from *main* (), when requested with the *exit* (), *_exit* (), or *_Exit* () functions; or when the last thread in the process terminates by returning from its start function, by calling the *pthread_exit* () function, or through cancellation. (2) Abnormal termination occurs when requested by the *abort* () function or when some signals are received.	**Note:** The *_exit* (), *_Exit* (), *abort* (), and *exit* () functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.

Clause	Source	Term	Definition	Notes
3. 298	POSIX	Process-To-Process Communication	The transfer of data between processes.	
3. 299	POSIX	Process Virtual Time	The measurement of time in units elapsed by the system clock while a process is executing.	
3. 300	POSIX	Program	A prepared sequence of instructions to the system to accomplish a defined task. The term "program" in IEEE Std 1003.1-2001 encompasses applications written in the Shell Command Language, complex utility input languages (for example, *awk* , *lex* , *sed* , and so on), and high-level languages.	
3. 301	POSIX	Protocol	A set of semantic and syntactic rules for exchanging information.	
3. 302	POSIX	Pseudo-Terminal	A facility that provides an interface that is identical to the terminal subsystem. A pseudo-terminal is composed of two devices: the "master device" and a "slave device". The slave device provides processes with an interface that is identical to the terminal interface, although there need not be hardware behind that interface. Anything written on the master device is presented to the slave as an input and anything written on the slave device is presented as an input on the master side.	
3. 303	POSIX	Radix Character	The character that separates the integer part of a number from the fractional part.	
3. 304	POSIX	Read-Only File System	A file system that has implementation-defined characteristics restricting modifications.	Note: File Times Update is described in detail in File Times Update.

Clause	Source	Term	Definition	Notes
3. 305	POSIX	Read-Write Lock	Multiple readers, single writer (read-write) locks allow many threads to have simultaneous read-only access to data while allowing only one thread to have write access at any given time. They are typically used to protect data that is read-only more frequently than it is changed. Read-write locks can be used to synchronize threads in the current process and other processes if they are allocated in memory that is writable and shared among the cooperating processes and have been initialized for this behavior.	
3. 306	POSIX	Real Group ID	The attribute of a process that, at the time of process creation, identifies the group of the user who created the process; see also Group ID.	
3. 307	POSIX	Real Time	Time measured as total units elapsed by the system clock without regard to which thread is executing.	
3. 308	POSIX	Realtime Signal Extension	A determinism improvement facility to enable asynchronous signal notifications to an application to be queued without impacting compatibility with the existing signal functions.	
3. 309	POSIX	Real User ID	The attribute of a process that, at the time of process creation, identifies the user who created the process; see also User ID.	
3. 310	POSIX	Record	A collection of related data units or words which is treated as a unit.	
3. 311	POSIX	Redirection	In the shell command language, a method of associating files with the input or output of commands.	Note: For further information, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.7, Redirection.

Clause	Source	Term	Definition	Notes
3. 312	POSIX	Redirection Operator	In the shell command language, a token that performs a redirection function. It is one of the following symbols: < > >| << >> <& >& <<- <>	
3. 313	POSIX	Reentrant Function	A function whose effect, when called by two or more threads, is guaranteed to be as if the threads each executed the function one after another in an undefined order, even if the actual execution is interleaved.	
3. 314	POSIX	Referenced Shared Memory Object	A shared memory object that is open or has one or more mappings defined on it.	
3. 315	POSIX	Refresh	To ensure that the information on the user's terminal screen is up-to-date.	
3. 316	POSIX	Regular Expression	A pattern that selects specific strings from a set of character strings.	Note: Regular Expressions are described in detail in Regular Expressions.
3. 317	POSIX	Region	In the context of the address space of a process, a sequence of addresses. In the context of a file, a sequence of offsets.	
3. 318	POSIX	Regular File	A file that is a randomly accessible sequence of bytes, with no further structure imposed by the system.	
3. 319	POSIX	Relative Pathname	A pathname not beginning with a slash.	Note: Pathname Resolution is defined in detail in Pathname Resolution.
3. 320	POSIX	Relocatable File	A file holding code or data suitable for linking with other object files to create an executable or a shared object file.	
3. 321	POSIX	Relocation	The process of connecting symbolic references with symbolic definitions. For example, when a program calls a function, the associated call instruction transfers control to the proper destination address at execution.	

Clause	Source	Term	Definition	Notes
3. 322	POSIX	Requested Batch Service	A service that is either rejected or performed prior to a response from the service to the requester.	
3. 323	POSIX	Resolution (of time)	The minimum time interval that a clock can measure or whose passage a timer can detect.	
3. 324	POSIX	Root Directory	A directory, associated with a process, that is used in pathname resolution for pathnames that begin with a slash.	
3. 325	POSIX	Runnable Process (or Thread)	A thread that is capable of being a running thread, but for which no processor is available.	
3. 326	POSIX	Running Process (or Thread)	A thread currently executing on a processor. On multi-processor systems there may be more than one such thread in a system at a time.	
3. 327	POSIX	Saved Resource Limits	An attribute of a process that provides some flexibility in the handling of unrepresentable resource limits, as described in the exec family of functions and setrlimit().	Note: The exec and setrlimit() functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 328	POSIX	Saved Set-Group-ID	An attribute of a process that allows some flexibility in the assignment of the effective group ID attribute, as described in the exec family of functions and setgid().	Note: The exec and setgid() functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 329	POSIX	Saved Set-User-ID	An attribute of a process that allows some flexibility in the assignment of the effective user ID attribute, as described in the exec family of functions and setuid().	Note: The exec and setuid() functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 330	POSIX	Scheduling	The application of a policy to select a runnable process or thread to become a running process or thread, or to alter one or more of the thread lists.	
3. 331	POSIX	Scheduling Allocation Domain	The set of processors on which an individual thread can be scheduled at any given time.	

Clause	Source	Term	Definition	Notes
3. 332	POSIX	Scheduling Contention Scope	A property of a thread that defines the set of threads against which that thread competes for resources. For example, in a scheduling decision, threads sharing scheduling contention scope compete for processor resources. In IEEE Std 1003.1-2001, a thread has scheduling contention scope of either PTHREAD_SCOPE_SYSTEM or PTHREAD_SCOPE_PROCESS.	
3. 333	POSIX	Scheduling Policy	A set of rules that is used to determine the order of execution of processes or threads to achieve some goal.	Note: Scheduling Policy is defined in detail in Scheduling Policy.
3. 334	POSIX	Screen	A rectangular region of columns and lines on a terminal display. A screen may be a portion of a physical display device or may occupy the entire physical area of the display device.	
3. 335	POSIX	Scroll	To move the representation of data vertically or horizontally relative to the terminal screen. There are two types of scrolling: (1) The cursor moves with the data, (2) The cursor remains stationary while the data moves.	
3. 336	POSIX	Semaphore	A minimum synchronization primitive to serve as a basis for more complex synchronization mechanisms to be defined by the application program.	Note: Semaphores are defined in detail in Semaphore.
3. 337	POSIX	Session	A collection of process groups established for job control purposes. Each process group is a member of a session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its creator. A process can alter its session membership; see setsid(). There can be multiple process groups in the same session.	Note: The setsid() function is defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.

Clause	Source	Term	Definition	Notes
3. 338	POSIX	Session Leader	A process that has created a session.	Note: For further information, see the setsid() function defined in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 339	POSIX	Session Lifetime	The period between when a session is created and the end of the lifetime of all the process groups that remain as members of the session.	
3. 340	POSIX	Shared Memory Object	An object that represents memory that can be mapped concurrently into the address space of more than one process.	
3. 341	POSIX	Shell	A program that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal.	
3. 342	POSIX	Shell, the	The Shell Command Language Interpreter; a specific instance of a shell.	Note: For further information, see the sh utility defined in the Shell and Utilities volume of IEEE Std 1003.1-2001.
3. 343	POSIX	Shell Script	A file containing shell commands. If the file is made executable, it can be executed by specifying its name as a simple command. Execution of a shell script causes a shell to execute the commands within the script. Alternatively, a shell can be requested to execute the commands in a shell script by specifying the name of the shell script as the operand to the sh utility.	**Note:** Simple Commands are defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.9.1, Simple Commands. The *sh* utility is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001.
3. 344	POSIX	Signal	A mechanism by which a process or thread may be notified of, or affected by, an event occurring in the system. Examples of such events include hardware exceptions and specific actions by processes. The term signal is also used to refer to the event itself.	
3. 345	POSIX	Signal Stack	Memory established for a thread, in which signal handlers catching signals sent to that thread are executed.	

Clause	Source	Term	Definition	Notes
3. 346	POSIX	Single-Quote	The character `'''`, also known as apostrophe.	
3. 347	POSIX	Slash	The character `'/'`, also known as solidus.	
3. 348	POSIX	Socket	A file of a particular type that is used as a communications endpoint for process-to-process communication as described in the System Interfaces volume of IEEE Std 1003.1-2001.	
3. 349	POSIX	Socket Address	An address associated with a socket or remote endpoint, including an address family identifier and addressing information specific to that address family. The address may include multiple parts, such as a network address associated with a host system and an identifier for a specific endpoint.	
3. 350	POSIX	Soft Limit	A resource limitation established for each process that the process may set to any value less than or equal to the hard limit.	
3. 351	POSIX	Source Code	When dealing with the Shell Command Language, input to the command language interpreter. The term "shell script" is synonymous with this meaning. When dealing with an ISO/IEC-conforming programming language, source code is input to a compiler conforming to that ISO/IEC standard. Source code also refers to the input statements prepared for the following standard utilities: *awk* , *bc* , *ed* , *lex* , *localedef* , *make* , *sed* , and *yacc* . Source code can also refer to a collection of sources meeting any or all of these meanings.	**Note:** The *awk* , *bc* , *ed* , *lex* , *localedef* , *make* , *sed* , and *yacc* utilities are defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001.

Clause	Source	Term	Definition	Notes
3. 352	POSIX	Space Character (<space>)	The character defined in the portable character set as <space>. The <space> is a member of the space character class of the current locale, but represents the single character, and not all of the possible members of the class; see also White Space.	
3. 353	POSIX	Spawn	A process creation primitive useful for systems that have difficulty with *fork* () and as an efficient replacement for *fork* ()/ *exec* .	
3. 354	POSIX	Special Built-In	See Built-In Utility in Built-In Utility (or Built-In).	
3. 355	POSIX	Special Parameter	In the shell command language, a parameter named by a single character from the following list: `*  @  #  ?  !  -  $  0`	Note: For further information, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.5.2, Special Parameters.
3. 356	POSIX	Spin Lock	A synchronization object used to allow multiple threads to serialize their access to shared data.	
3. 357	POSIX	Sporadic Server	A scheduling policy for threads and processes that reserves a certain amount of execution capacity for processing aperiodic events at a given priority level.	
3. 358	POSIX	Standard Error	An output stream usually intended to be used for diagnostic messages.	
3. 359	POSIX	Standard Input	An input stream usually intended to be used for primary data input.	
3. 360	POSIX	Standard Output	An output stream usually intended to be used for primary data output.	
3. 361	POSIX	Standard Utilities	The utilities described in the Shell and Utilities volume of IEEE Std 1003.1-2001.	

Clause	Source	Term	Definition	Notes
3. 362	POSIX	Stream	Appearing in lowercase, a stream is a file access object that allows access to an ordered sequence of characters, as described by the ISO C standard. Such objects can be created by the *fdopen* (), *fopen* (), or *popen* () functions, and are associated with a file descriptor. A stream provides the additional services of user-selectable buffering and formatted input and output; see also STREAM.	**Note:** For further information, see the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.5, Standard I/O Streams. The *fdopen* (), *fopen* (), or *popen* () functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 363	POSIX	STREAM	Appearing in uppercase, STREAM refers to a full-duplex connection between a process and an open device or pseudo-device. It optionally includes one or more intermediate processing modules that are interposed between the process end of the STREAM and the device driver (or pseudo-device driver) end of the STREAM; see also Stream.	Note: For further information, see the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.6, STREAMS.
3. 364	POSIX	STREAM End	The STREAM end is the driver end of the STREAM and is also known as the downstream end of the STREAM.	
3. 365	POSIX	STREAM Head	The STREAM head is the beginning of the STREAM and is at the boundary between the system and the application process. This is also known as the upstream end of the STREAM.	
3. 366	POSIX	STREAMS Multiplexor	A driver with multiple STREAMS connected to it. Multiplexing with STREAMS connected above is referred to as N-to-1, or "upper multiplexing". Multiplexing with STREAMS connected below is referred to as 1-to-N or "lower multiplexing".	
3. 367	POSIX	String	A contiguous sequence of bytes terminated by and including the first null byte.	

Clause	Source	Term	Definition	Notes
3. 368	POSIX	Subshell	A shell execution environment, distinguished from the main or current shell execution environment.	Note: For further information, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.12, Shell Execution Environment.
3. 369	POSIX	Successfully Transferred	For a write operation to a regular file, when the system ensures that all data written is readable on any subsequent open of the file (even one that follows a system or power failure) in the absence of a failure of the physical storage medium. For a read operation, when an image of the data on the physical storage medium is available to the requesting process.	
3. 370	POSIX	Supplementary Group ID	An attribute of a process used in determining file access permissions. A process has up to {NGROUPS_MAX} supplementary group IDs in addition to the effective group ID. The supplementary group IDs of a process are set to the supplementary group IDs of the parent process when the process is created.	
3. 371	POSIX	Suspended Job	A job that has received a SIGSTOP, SIGTSTP, SIGTTIN, or SIGTTOU signal that caused the process group to stop. A suspended job is a background job, but a background job is not necessarily a suspended job.	
3. 372	POSIX	Symbolic Link	A type of file with the property that when the file is encountered during pathname resolution, a string stored by the file is used to modify the pathname resolution. The stored string has a length of {SYMLINK_MAX} bytes or fewer.	Note: Pathname Resolution is defined in detail in Pathname Resolution.

Clause	Source	Term	Definition	Notes
3. 373	POSIX	Synchronized Input and Output	A determinism and robustness improvement mechanism to enhance the data input and output mechanisms, so that an application can ensure that the data being manipulated is physically present on secondary mass storage devices.	
3. 374	POSIX	Synchronized I/O Completion	The state of an I/O operation that has either been successfully transferred or diagnosed as unsuccessful.	
3. 375	POSIX	Synchronized I/O Data Integrity Completion	For read, when the operation has been completed or diagnosed if unsuccessful. The read is complete only when an image of the data has been successfully transferred to the requesting process. If there were any pending write requests affecting the data to be read at the time that the synchronized read operation was requested, these write requests are successfully transferred prior to reading the data. For write, when the operation has been completed or diagnosed if unsuccessful. The write is complete only when the data specified in the write request is successfully transferred and all file system information required to retrieve the data is successfully transferred. File attributes that are not necessary for data retrieval (access time, modification time, status change time) need not be successfully transferred prior to returning to the calling process.	

Clause	Source	Term	Definition	Notes
3. 376	POSIX	Synchronized I/O File Integrity Completion	Identical to a synchronized I/O data integrity completion with the addition that all file attributes relative to the I/O operation (including access time, modification time, status change time) are successfully transferred prior to returning to the calling process.	
3. 377	POSIX	Synchronized I/O Operation	An I/O operation performed on a file that provides the application assurance of the integrity of its data and files.	
3. 378	POSIX	Synchronous I/O Operation	An I/O operation that causes the thread requesting the I/O to be blocked from further use of the processor until that I/O operation completes.	**Note:** A synchronous I/O operation does not imply synchronized I/O data integrity completion or synchronized I/O file integrity completion.
3. 379	POSIX	Synchronously-Generated Signal	A signal that is attributable to a specific thread.	For example, a thread executing an illegal instruction or touching invalid memory causes a synchronously-generated signal. Being synchronous is a property of how the signal was generated and not a property of the signal number.
3. 380	POSIX	System	An implementation of IEEE Std 1003.1-2001.	
3. 381	POSIX	System Crash	An interval initiated by an unspecified circumstance that causes all processes (possibly other than special system processes) to be terminated in an undefined manner, after which any changes to the state and contents of files created or written to by an application prior to the interval are undefined, except as required elsewhere in IEEE Std 1003.1-2001.	

Clause	Source	Term	Definition	Notes
3. 382	POSIX	System Console	A device that receives messages sent by the *syslog* () function, and the *fmtmsg* () function when the MM_CONSOLE flag is set.	**Note:** The *syslog* () and *fmtmsg* () functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 383	POSIX	System Databases	An implementation provides two system databases: the "group database" (see also Group Database) and the "user database" (see also User Database).	
3. 384	POSIX	System Documentation	All documentation provided with an implementation except for the conformance document. Electronically distributed documents for an implementation are considered part of the system documentation.	
3. 385	POSIX	System Process	An object other than a process executing an application, that is provided by the system and has a process ID.	
3. 386	POSIX	System Reboot	An unspecified sequence of events that may result in the loss of transitory data; that is, data that is not saved in permanent storage. For example, message queues, shared memory, semaphores, and processes.	
3. 387	POSIX	System Trace Event	A trace event that is generated by the implementation, in response either to a system-initiated action or to an application-requested action, except for a call to posix_trace_event(). When supported by the implementation, a system-initiated action generates a process-independent system trace event and an application-requested action generates a process-dependent system trace event. For a system trace event not defined by IEEE Std 1003.1-2001, the associated trace event type identifier is derived from the implementation-defined name for this trace event, and the associated data is of implementation-defined content and length.	

Clause	Source	Term	Definition	Notes
3. 388	POSIX	System-Wide	Pertaining to events occurring in all processes existing in an implementation at a given point in time.	
3. 389	POSIX	Tab Character (<tab>)	A character that in the output stream indicates that printing or displaying should start at the next horizontal tabulation position on the current line. It is the character designated by '\t' in the C language. If the current position is at or past the last defined horizontal tabulation position, the behavior is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the tabulation.	
3. 390	POSIX	Terminal (or Terminal Device)	A character special file that obeys the specifications of the general terminal interface.	Note: The General Terminal Interface is defined in detail in General Terminal Interface .
3. 391	POSIX	Text Column	A roughly rectangular block of characters capable of being laid out side-by-side next to other text columns on an output page or terminal screen. The widths of text columns are measured in column positions.	
3. 392	POSIX	Text File	A file that contains characters organized into one or more lines. The lines do not contain NUL characters and none can exceed {LINE_MAX} bytes in length, including the <newline>. Although IEEE Std 1003.1-2001 does not distinguish between text files and binary files (see the ISO C standard), many utilities only produce predictable or meaningful output when operating on text files. The standard utilities that have such restrictions always specify "text files" in their STDIN or INPUT FILES sections.	

Clause	Source	Term	Definition	Notes
3. 393	POSIX	Thread	A single flow of control within a process. Each thread has its own thread ID, scheduling priority and policy, errno value, thread-specific key/value bindings, and the required system resources to support a flow of control. Anything whose address may be determined by a thread, including but not limited to static variables, storage obtained via malloc(), directly addressable storage obtained through implementation-defined functions, and automatic variables, are accessible to all threads in the same process.	Note: The malloc() function is defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 394	POSIX	Thread ID	Each thread in a process is uniquely identified during its lifetime by a value of type **pthread_t** called a thread ID.	
3. 395	POSIX	Thread List	An ordered set of runnable threads that all have the same ordinal value for their priority. The ordering of threads on the list is determined by a scheduling policy or policies. The set of thread lists includes all runnable threads in the system.	
3. 396	POSIX	Thread-Safe	A function that may be safely invoked concurrently by multiple threads. Each function defined in the System Interfaces volume of IEEE Std 1003.1-2001 is thread-safe unless explicitly stated otherwise. Examples are any "pure" function, a function which holds a mutex locked while it is accessing static storage, or objects shared among threads.	

Clause	Source	Term	Definition	Notes
3. 397	POSIX	Thread-Specific Data Key	A process global handle of type **pthread_key_t** which is used for naming thread-specific data. Although the same key value may be used by different threads, the values bound to the key by *pthread_setspecific* () and accessed by *pthread_getspecific* () are maintained on a per-thread basis and persist for the life of the calling thread.	**Note:** The *pthread_getspecific* () and *pthread_setspecific* () functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 398	POSIX	Tilde	The character '~'.	
3. 399	POSIX	Timeouts	A method of limiting the length of time an interface will block; see also Blocked Process (or Thread) .	
3. 400	POSIX	Timer	A mechanism that can notify a thread when the time as measured by a particular clock has reached or passed a specified value, or when a specified amount of time has passed.	
3. 401	POSIX	Timer Overrun	A condition that occurs each time a timer, for which there is already an expiration signal queued to the process, expires.	
3. 402	POSIX	Token	In the shell command language, a sequence of characters that the shell considers as a single unit when reading input. A token is either an operator or a word.	Note: The rules for reading input are defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.3, Token Recognition.
3. 403	POSIX	Trace Analyzer Process	A process that extracts trace events from a trace stream to retrieve information about the behavior of an application.	
3. 404	POSIX	Trace Controller Process	A process that creates a trace stream for tracing a process.	
3. 405	POSIX	Trace Event	A data object that represents an action executed by the system, and that is recorded in a trace stream.	
3. 406	POSIX	Trace Event Type	A data object type that defines a class of trace event.	

Clause	Source	Term	Definition	Notes
3. 407	POSIX	Trace Event Type Mapping	A one-to-one mapping between trace event types and trace event names.	
3. 408	POSIX	Trace Filter	A filter that allows the trace controller process to specify those trace event types that are to be ignored; that is, not generated.	
3. 409	POSIX	Trace Generation Version	A data object that is an implementation-defined character string, generated by the trace system and describing the origin and version of the trace system.	
3. 410	POSIX	Trace Log	The flushed image of a trace stream, if the trace stream is created with a trace log.	
3. 411	POSIX	Trace Point	An action that may cause a trace event to be generated.	
3. 412	POSIX	Trace Stream	An opaque object that contains trace events plus internal data needed to interpret those trace events.	
3. 413	POSIX	Trace Stream Identifier	A handle to manage tracing operations in a trace stream.	
3. 414	POSIX	Trace System	A system that allows both system and user trace events to be generated into a trace stream. These trace events can be retrieved later.	
3. 415	POSIX	Traced Process	A process for which at least one trace stream has been created. A traced process is also called a target process.	
3. 416	POSIX	Tracing Status of a Trace Stream	A status that describes the state of an active trace stream. The tracing status of a trace stream can be retrieved from the trace stream attributes. An active trace stream can be in one of two states: running or suspended.	

Clause	Source	Term	Definition	Notes
3. 417	POSIX	Typed Memory Name Space	A system-wide name space that contains the names of the typed memory objects present in the system. It is configurable for a given implementation.	
3. 418	POSIX	Typed Memory Object	A combination of a typed memory pool and a typed memory port. The entire contents of the pool are accessible from the port. The typed memory object is identified through a name that belongs to the typed memory name space.	
3. 419	POSIX	Typed Memory Pool	An extent of memory with the same operational characteristics. Typed memory pools may be contained within each other.	
3. 420	POSIX	Typed Memory Port	A hardware access path to one or more typed memory pools.	
3. 421	POSIX	Unbind	Remove the association between a network address and an endpoint.	
3. 422	POSIX	Unit Data	See Datagram in Datagram.	
3. 423	POSIX	Upshifting	The conversion of a lowercase character that has a single-character uppercase representation into this uppercase representation.	
3. 424	POSIX	User Database	A system database that contains at least the following information for each user ID: (1) User name, (2) Numerical user ID, (3) Initial numerical group ID, (4) Initial working directory, (5) Initial user program. The initial numerical group ID is used by the newgrp utility. Any other circumstances under which the initial values are operative are implementation-defined. If the initial user program field is null, an implementation-defined program is used. If the initial working directory field is null, the interpretation of that field is implementation-defined.	Note: The newgrp utility is defined in detail in the Shell and Utilities volume of IEEE Std 1003.1-2001.

Clause	Source	Term	Definition	Notes
3. 425	POSIX	User ID	A non-negative integer that is used to identify a system user. When the identity of a user is associated with a process, a user ID value is referred to as a real user ID, an effective user ID, or a saved set-user-ID.	
3. 426	POSIX	User Name	A string that is used to identify a user; see also User Database. To be portable across systems conforming to IEEE Std 1003.1-2001, the value is composed of characters from the portable filename character set. The hyphen should not be used as the first character of a portable user name.	
3. 427	POSIX	User Trace Event	A trace event that is generated explicitly by the application as a result of a call to posix_trace_event().	
3. 428	POSIX	Utility	A program, excluding special built-in utilities provided as part of the Shell Command Language, that can be called by name from a shell to perform a specific task, or related set of tasks.	Note: For further information on special built-in utilities, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.14, Special Built-In Utilities.
3. 429	POSIX	Variable	In the shell command language, a named parameter.	Note: For further information, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.5, Parameters and Variables.
3. 430	POSIX	Vertical-Tab Character (<vertical-tab>)	A character that in the output stream indicates that printing should start at the next vertical tabulation position. It is the character designated by '\v' in the C language. If the current position is at or past the last defined vertical tabulation position, the behavior is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the tabulation.	

Clause	Source	Term	Definition	Notes
3. 431	POSIX	White Space	A sequence of one or more characters that belong to the **space** character class as defined via the *LC_CTYPE* category in the current locale. In the POSIX locale, white space consists of one or more <blank>s (<space>s and <tab>s), <newline>s, <carriage-return>s, <form-feed>s, and <vertical-tab>s.	
3. 432	POSIX	Wide-Character Code (C Language)	An integer value corresponding to a single graphic symbol or control code.	Note: C Language Wide-Character Codes are defined in detail in C Language Wide-Character Codes.
3. 433	POSIX	Wide-Character Input/Output Functions	The functions that perform wide-oriented input from streams or wide-oriented output to streams: *fgetwc* (), *fgetws* (), *fputwc* (), *fputws* (), *fwprintf* (), *fwscanf* (), *getwc* (), *getwchar* (), *putwc* (), *putwchar* (), *ungetwc* (), *vfwprintf* (), *vfwscanf* (), *vwprintf* (), *vwscanf* (), *wprintf* (), and *wscanf* ().	**Note:** These functions are defined in detail in the System Interfaces volume of IEEE Std 1003.1-2001.
3. 434	POSIX	Wide-Character String	A contiguous sequence of wide-character codes terminated by and including the first null wide-character code.	
3. 435	POSIX	Word	In the shell command language, a token other than an operator. In some cases a word is also a portion of a word token: in the various forms of parameter expansion, such as ${*name* -*word* }, and variable assignment, such as *name* =*word* , the word is the portion of the token depicted by *word* . The concept of a word is no longer applicable following word expansions-only fields remain.	**Note:** For further information, see the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6.2, Parameter Expansion and the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.6, Word Expansions.

Clause	Source	Term	Definition	Notes
3. 436	POSIX	Working Directory (or Current Working Directory)	A directory, associated with a process, that is used in pathname resolution for pathnames that do not begin with a slash.	
3. 437	POSIX	Worldwide Portability Interface	Functions for handling characters in a codeset-independent manner.	
3. 438	POSIX	Write	To output characters to a file, such as standard output or standard error. Unless otherwise stated, standard output is the default output destination for all uses of the term "write"; see the distinction between display and write in Display.	
3. 439	POSIX	XSI	The X/Open System Interface is the core application programming interface for C and sh programming for systems conforming to the Single UNIX Specification. This is a superset of the mandatory requirements for conformance to IEEE Std 1003.1-2001.	
3. 440	POSIX	XSI-Conformant	A system which allows an application to be built using a set of services that are consistent across all systems that conform to IEEE Std 1003.1-2001 and that support the XSI extension.	Note: See also Conformance.
3. 441	POSIX	Zombie Process	A process that has terminated and that is deleted when its exit status has been reported to another process which is waiting for that process to terminate.	
3. 442	POSIX	±0	The algebraic sign provides additional information about any variable that has the value zero when the representation allows the sign to be determined.	
1.7. 1	ISLisp	abstract class	A class that by definition has no direct instances.	

Clause	Source	Term	Definition	Notes
1.7. 2	ISLisp	activation	Computation of a function. Every activation has an activation point, an activation period, and an activation end. The activator, which is a function application form prepared for execution, starts the activation at the activation point.	
1.7. 3	ISLisp	accessor	Association of a reader and a writer for a slot of an instance.	
1.7. 4	ISLisp	binding	Binding has both a syntactic and a semantic aspect. xxx Syntactically, ``binding'' describes the relation between an identifier and a binding ISLisp form. The property of being bound can be checked textually by relating defining and applied identifier occurrences. xxx Semantically, ``binding'' describes the relation between a variable, its denoting identifier, and an object (or, the relation between a variable and a location). This relation might be imagined to be materialized in some entity, the binding. Such a binding entity is constructed at run time and destroyed later, or might have indefinite extent.	
1.7. 5	ISLisp	class	Object, that determines the structure and behavior of a set of other objects, called its instances. The behavior is the set of operations that can be performed on an instance.	
1.7. 6	ISLisp	condition	An object that represents a situation that has been (or might be) detected by a running program.	
1.7. 7	ISLisp	definition point	An identifier represents an ISLisp object starting with its definition point, which is a textual point of an ISLisp text.	
1.7. 8	ISLisp	direct instance	Every ISLisp object is direct instance of exactly one class, which is called ``its class''. The set of all direct instances together with their behavior constitute a class.	
1.7. 9	ISLisp	dynamic	Having an effect that is determined only through program execution and that cannot, in general, be determined statically.	

Clause	Source	Term	Definition	Notes
1.7. 10	ISLisp	dynamic variable	A variable whose associated binding is determined by the most recently executed active block that established it, rather than statically by a lexically apparent block according to the lexical principle.	
1.7. 11	ISLisp	evaluation	Computation of a form prepared for execution which results in a value and/or a side effect.	
1.7. 12	ISLisp	execution	A sequence of (sometimes nested) activations.	
1.7. 13	ISLisp	extension	An implementation defined modification to the requirements of this International Standard that does not invalidate any ISLisp text complying with this International Standard (except by prohibiting the use of one or more particular spellings of identifiers), does not alter the set of actions which are required to signal errors, and does not alter the status of any feature designated as implementation dependent.	
1.7. 14	ISLisp	form	A single, syntactically valid unit of program text, capable of being prepared for execution.	
1.7. 15	ISLisp	function	An ISLisp object that is called with arguments, performs a computation (possibly having side-effects), and returns a value.	
1.7. 16	ISLisp	generic function	Function whose application behavior is determined by the classes of the values of its arguments and which consists -- in general -- of several methods.	
1.7. 17	ISLisp	identifier	A lexical element (lexeme) which designates an ISLisp object. In the data structure representation of ISLisp texts, identifiers are denoted by symbols.	
1.7. 18	ISLisp	immutable binding	A binding is immutable if the relation between an identifier and the object represented by this identifier cannot be changed. It is a violation if there is attempt to change an immutable binding (error-id. *immutable-binding* .	

Clause	Source	Term	Definition	Notes
1.7. 19	ISLisp	immutable object	An object is immutable if it is not subject to change, either because no operator is provided that is capable of effecting such change, or because some constraint exists which prohibits the use of an operator that might otherwise be capable of effecting such a change. Except as explicitly indicated otherwise, a conforming processor is not required to detect attempts to modify immutable objects; the consequences are undefined if an attempt is made to modify an immutable object.	
1.7. 20	ISLisp	implementation defined	A feature, possibly differing between different ISLisp processors, but completely defined for every processor.	
1.7. 21	ISLisp	implementation dependent	A feature, possibly differing between different ISLisp processors, but not necessarily defined for any particular processor.	Note - A conforming ISLisp text must not depend upon implementation dependent features.
1.7. 22	ISLisp	inheritance	Relation between a class and its superclass which maps structure and behavior of the superclass onto the class. ISLisp supports a restricted form of multiple inheritance; *i.e* ., a class may have several direct superclasses at once.	
1.7. 23	ISLisp	instance (of a class)	Either a direct instance of a class or an instance of one of its subclasses.	
1.7. 24	ISLisp	literal	An object whose representation occurs directly in a program as a constant value.	
1.7. 25	ISLisp	metaclass	A class whose instances are themselves classes.	
1.7. 26	ISLisp	method	Case of a generic function for a particular parameter profile, which defines the class-specific behavior and operations of the generic function.	
1.7. 27	ISLisp	object	An object is anything that can be created, destroyed, manipulated, compared, stored, input, or output by the ISLisp processor. In particular, functions are ISLisp objects. Objects that can be passed as arguments to functions, can be returned as values, can be bound to variables, and can be part of structures, are called *first-class objects* .	

Clause	Source	Term	Definition	Notes
1.7. 28	ISLisp	operator	The first element of a compound form, which is either a reserved name that identifies the form as a special form, or the name of a macro, or a lambda expression, or else an identifier in the function namespace.	
1.7. 29	ISLisp	parameter profile	Parameter list of a method, where each formal parameter is accompanied by its class name. If a parameter is not accompanied by a class name, it belongs to the most general class.	
1.7. 30	ISLisp	place	Objects can be stored in places and retrieved later. Places are designated by forms which are permitted as the first argument of `setf`. If used this way an object is stored in the place. If the form is not used as first argument of setf the stored object is retrieved. The cases are listed in the description of `setf`.	
1.7. 31	ISLisp	position	(a) argument position: Occurrence of a text unit as an element in a form excluding the first one. xxx(b) operator position: Occurrence of a text unit as the first element in a form.	
1.7. 32	ISLisp	process	The execution of an ISLisp text prepared for execution.	
1.7. 33	ISLisp	processor	A system or mechanism, that accepts an ISLisp text (or an equivalent data structure) as input, prepares it for execution, and executes the result to produce values and side effects.	
1.7. 34	ISLisp	program	An aggregation of expressions to be evaluated, the specific nature of which depends on context. Within this International Standard, the term ``program'' is used only in an abstract way; there is no specific syntactic construct that delineates a program.	
1.7. 35	ISLisp	scope	The scope of an identifier is that textual part of a program where the meaning of that identifier is defined; *i.e* ., there exists an ISLisp object designated by this identifier.	
1.7. 36	ISLisp	slot	A named component of an instance which can be accessed using the slot accessors. The structure of an instance is defined by the set of its slots.	

Clause	Source	Term	Definition	Notes
1.7. 37	ISLisp	text	A text that complies with the requirements of this International Standard (*i.e.*, with the syntax and static semantics of ISLisp). An ISLisp text consists of a sequence of toplevel forms.	
1.7. 38	ISLisp	toplevel form	Any form that either is not nested in any other form or is nested only in `progn` forms.	
1.7. 39	ISLisp	toplevel scope	The scope in which a complete ISLisp text unit is processed.	
1.7. 40	ISLisp	writer	A method associated with a slot of a class, whose task is to bind a value with a slot of an instance of that class.	
4. 1	Cobol	absolute item	An item in a report that has a fixed position on a page.	
4. 2	Cobol	activated runtime element	A function, method, or program placed into the active state.	
4. 3	Cobol	activating statement	A statement that causes the execution of a function, method, or program.	
4. 4	Cobol	activating runtime element	The function, method, or program that executed a given activating statement.	
4. 5	Cobol	active state	The state of a function, method, or program that has been activated but has not yet returned control to the activating runtime element.	
4. 6	Cobol	alphabetic character (in the COBOL character repertoire)	A basic letter or a space character.	
4. 7	Cobol	alphanumeric character	Any coded character in an alphanumeric coded character set, whether or not there is an assigned graphic symbol for that coded character.	

Clause	Source	Term	Definition	Notes
4. 8	Cobol	alphanumeric character position	The amount of physical storage required to store, or presentation space required to print or display, a single character of an alphanumeric character set.	
4. 9	Cobol	alphanumeric character set; alphanumeric coded character set	See alphanumeric coded character set.	
4. 10	Cobol	alphanumeric coded character set; alphanumeric character set	A coded character set that the implementor has designated for representation of data items of usage display and alphanumeric literals.	
4. 11	Cobol	alphanumeric group item	Any group item except for: (1) a strongly-typed group item, (2) a bit group item, (3) a national group item.	
4. 12	Cobol	argument	An operand specified in an activating statement that specifies the data to be passed.	
4. 13	Cobol	assumed decimal point	A decimal point position that does not involve the existence of an actual character in a data item. An assumed decimal point has logical meaning with no physical representation.	
4. 14	Cobol	based data item	A data item established by association of a based entry with an actual data item or allocated storage.	
4. 15	Cobol	based entry	A data description entry that serves as a template that is dynamically associated with data items or allocated storage.	

Clause	Source	Term	Definition	Notes
4. 16	Cobol	basic letter	One of the uppercase letters 'A' through 'Z' and the lowercase letters 'a' through 'z' in the COBOL character repertoire.	
4. 17	Cobol	bit	The smallest unit in a computer's storage structure capable of representing two distinct alternatives.	
4. 18	Cobol	bit data item	An elementary data item of category boolean and usage bit or a bit group item.	
4. 19	Cobol	block; physical record	A physical unit of data that is normally composed of one or more logical records.	
4. 20	Cobol	boolean character	A unit of information that consists of the value zero or one. Each boolean character may be represented in storage as a bit, an alphanumeric character, or a national character.	
4. 21	Cobol	boolean data item	A data item capable of containing a boolean value.	
4. 22	Cobol	boolean expression	One or more boolean operands separated by boolean operators.	
4. 23	Cobol	boolean position	The amount of physical storage required to store, or presentation space required to print or display, a single boolean character.	
4. 24	Cobol	boolean value	A value consisting of a sequence of one or more boolean characters.	
4. 25	Cobol	byte	A sequence of bits representing the smallest addressable character unit in the memory of a given computer.	

Clause	Source	Term	Definition	Notes
4. 26	Cobol	character (in a coded character set)	A code value that constitutes the coded representation of a letter, digit, symbol, control function, or other member of a set of elements used for the organization, control, or representation of data.	NOTE COBOL processes each code value in a coded character set as though it were a character.
4. 27	Cobol	character (in a screen item)	A graphic character.	
4. 28	Cobol	character (in COBOL character repertoire); COBOL character	A letter, digit, or special character, independent of its coded representation, that is used in formation of the words and separators of COBOL.	
4. 29	Cobol	character (in computer storage)	A single code value of a coded character set.	
4. 30	Cobol	character boundary	The leftmost bit of an addressing boundary in the storage of the computer.	
4. 31	Cobol	character position	The amount of physical storage required to store, or presentation space required to print or display, a single character -- either an alphanumeric character or a national character. One element of a coded character set occupies one character position.	NOTE As an example, each element of a combining sequence in the UCS occupies one character position. For a UTF-16 surrogate value, the left surrogate occupies one character position and the right surrogate occupies another character position.
4. 32	Cobol	character-string	A sequence of contiguous characters that form a COBOL word, a literal, or a picture character-string.	

Clause	Source	Term	Definition	Notes
4. 33	Cobol	class (in object orientation)	The entity that defines common behavior and implementation for zero, one, or more objects.	
4. 34	Cobol	class (of a data item)	A designation for a set of data items having common attributes or a common range of values, defined by the PICTURE clause, the USAGE clause, or the PICTURE and USAGE clauses in a data description entry; by the definition of a predefined identifier; or by the definition of an intrinsic function.	
4. 35	Cobol	class (of data values)	A designation for a set of data values that are permissible in the content of a data item.	
4. 36	Cobol	class definition (in object orientation)	A compilation unit that defines a class of objects.	
4. 37	Cobol	clause	An ordered set of consecutive COBOL character-strings whose purpose is to specify an attribute of an entry.	
4. 38	Cobol	COBOL character; character (in COBOL character repertoire)	See character (in COBOL character repertoire).	
4. 39	Cobol	COBOL character repertoire	The repertoire of characters used in writing the syntax of a COBOL compilation group, except for comments and the content of non-hexadecimal alphanumeric and national literals.	

Clause	Source	Term	Definition	Notes
4. 40	Cobol	coded character set	A set of unambiguous rules that establishes a character set and the relationship between the characters of the set and their coded representation. [ISO/IEC 10646-1]	
4. 41	Cobol	combining character	A UCS member that is intended for combination with the preceding non-combining graphic character or with a sequence of combining characters preceded by a non-combining character.	
4. 42	Cobol	common program	A program that, despite being directly contained within another program, may be called from any program directly or indirectly contained in that other program.	
4. 43	Cobol	compilation group	A sequence of one or more compilation units submitted together for compilation.	
4. 44	Cobol	compilation unit	A source unit that is not nested within another source unit.	
4. 45	Cobol	composite sequence	A sequence of graphic characters consisting of a non-combining character followed by one or more combining characters. [ISO/IEC 10646-1].	
4. 46	Cobol	conditional statement	A statement for which the truth value of a specified condition is evaluated and used to determine subsequent flow of control.	
4. 47	Cobol	conformance (for object orientation)	A unidirectional relation that allows an object to be used according to an interface other than the interface of its own class.	

Clause	Source	Term	Definition	Notes
4. 48	Cobol	conformance (for parameters)	The requirements for the relationship between arguments and formal parameters and between returning items in activating and activated runtime elements.	
4. 49	Cobol	control function	An action that affects the recording, processing, transmission, or interpretation of data, and that has a coded representation consisting of one or more bytes.	NOTE This definition is the same as that in ISO/IEC 10646-1 except that "octets" is replaced by "bytes" because the term "octet" is not used in the COBOL specification.
4. 50	Cobol	cultural element	An element of data for computer use that may vary dependent on language, geographical territory, or other cultural circumstances.	
4. 51	Cobol	currency sign	The COBOL character '$', used as the default currency symbol in a picture character-string and as the default currency string that appears in the edited format of data items.	NOTE Features exist for selection of other currency strings and currency symbols.
4. 52	Cobol	currency string	The set of characters to be placed into numeric-edited data items as a result of editing operations when the item includes a currency symbol in its picture character string.	
4. 53	Cobol	currency symbol	The character used in a picture character-string to represent the presence of a currency string.	
4. 54	Cobol	current record	The record that is available in the record area associated with a file.	
4. 55	Cobol	current volume pointer	A conceptual entity that points to the current volume of a sequential file.	

Clause	Source	Term	Definition	Notes
4. 56	Cobol	data item	A unit of data defined by a data description entry or resulting from the evaluation of an identifier.	
4. 57	Cobol	debugging line	A source line that is optionally compiled, depending on the setting of a debugging mode switch.	
4. 58	Cobol	decimal point; decimal separator	The character used to represent the radix point. The default is the character period.	
4. 59	Cobol	decimal separator; decimal point	See decimal point.	
4. 60	Cobol	declarative statement	A USE statement, which defines the conditions under which the procedures that follow the statement will be executed.	
4. 61	Cobol	de-editing	The logical removal of all editing characters from a numeric-edited data item in order to determine that item's unedited numeric value.	
4. 62	Cobol	delimited scope statement	Any statement that is terminated by its explicit scope terminator.	
4. 63	Cobol	digit position	The amount of physical storage required to store, or presentation space required to print or display, a single digit.	
4. 64	Cobol	dynamic access	An access mode in which specific logical records may be obtained from or placed into a mass storage file in a nonsequential manner and obtained from a file in a sequential manner.	

Clause	Source	Term	Definition	Notes
4. 65	Cobol	dynamic storage	Storage that is allocated and released on request during runtime.	
4. 66	Cobol	end marker	A marker for the end of a source unit.	
4. 67	Cobol	entry	A descriptive set of consecutive clauses terminated by a separator period.	
4. 68	Cobol	entry convention	The information required to interact successfully with a function, method, or program.	
4. 69	Cobol	exception condition	A condition detected at runtime that indicates that an error or exception to normal processing has occurred.	
4. 70	Cobol	exception object	An object that acts as an exception condition.	
4. 71	Cobol	exception status indicator	A conceptual entity that exists for each exception-name.	
4. 72	Cobol	EXIT FUNCTION statement	an abbreviation for 'EXIT statement with the FUNCTION phrase'.	
4. 73	Cobol	EXIT METHOD statement	an abbreviation for 'EXIT statement with the METHOD phrase'.	
4. 74	Cobol	EXIT PARAGRAPH statement	an abbreviation for 'EXIT statement with the PARAGRAPH phrase'.	
4. 75	Cobol	EXIT PERFORM statement	an abbreviation for 'EXIT statement with the PERFORM phrase'.	
4. 76	Cobol	EXIT PROGRAM statement	an abbreviation for 'EXIT statement with the PROGRAM phrase'.	
4. 77	Cobol	EXIT SECTION statement	an abbreviation for 'EXIT statement with the SECTION phrase'.	
4. 78	Cobol	explicit scope terminator	A statement-dependent word that by its presence terminates the scope of that statement.	

Clause	Source	Term	Definition	Notes
4. 79	Cobol	extend mode	A mode of file processing in which records may be added at the end of a sequential file, but no records may be deleted, read, or updated.	
4. 80	Cobol	extended letter	A letter, other than the basic letters, in the set of characters defined for the COBOL character repertoire.	
4. 81	Cobol	external data	Data that belongs to the run unit and may be accessed by any runtime element in which it is described.	
4. 82	Cobol	external media format	A form of data suitable for presentation or printing, including any control functions necessary for representation as readable text.	
4. 83	Cobol	external switch	A hardware or software device, defined and named by the implementor, that is used to indicate that one of two alternate states exists.	
4. 84	Cobol	factory object	The single object associated with a class, defined by the factory definition of that class, typically used to create the instance objects of the class.	
4. 85	Cobol	file	A logical entity that represents a collection of logical records. There is one logical file associated with one file connector and there may be several logical files associated with one physical file.	

Clause	Source	Term	Definition	Notes
4. 86	Cobol	file connector	A storage area that contains information about a file and is used as the linkage between a file-name and a physical file and between a file-name and its associated record area.	
4. 87	Cobol	file organization	The permanent logical file structure established at the time that a file is created.	
4. 88	Cobol	file position indicator	A conceptual entity that either is used to facilitate exact specification of the next record to be accessed, or indicates why such a reference cannot be established.	
4. 89	Cobol	file sharing	A cooperative environment that controls concurrent access to the same physical file.	
4. 90	Cobol	fixed file attribute	An attribute of a physical file that is established when the physical file is created and is never changed during the existence of the physical file.	
4. 91	Cobol	formal parameter	A data-name specified in the USING phrase of the procedure division header that gives the name used in the function, method, or program for a parameter.	
4. 92	Cobol	function	An intrinsic or user-defined procedural entity that returns a value based upon the arguments.	

Clause	Source	Term	Definition	Notes
4. 93	Cobol	function prototype definition	A definition that specifies the rules governing the arguments needed for the evaluation of a particular function, the data item resulting from the evaluation of the function, and all other requirements needed for the evaluation of that function.	
4. 94	Cobol	graphic character	A character, other than a control function, that has a visual representation normally handwritten, printed, or displayed. [ISO/IEC 10646-1].	
4. 95	Cobol	graphic symbol	The visual representation of a graphic character or of a composite sequence. [ISO/IEC 10646-1].	
4. 96	Cobol	grouping (in locale editing)	The separation of digits into groups in number and currency formatting.	
4. 97	Cobol	grouping separator	The character used to separate digits in numbers for ease of reading. The default is the character comma.	
4. 98	Cobol	high-order end	The leftmost position of a string of characters or a string of bits.	
4. 99	Cobol	i-o mode	A mode of file processing in which records can be read, updated, added, and deleted.	
4. 100	Cobol	i-o status	A conceptual entity that exists for a file, that contains a value indicating the result of the execution of an input-output operation for that file.	
4. 101	Cobol	imperative statement	A statement that specifies an unconditional action or that is a delimited scope statement.	

Clause	Source	Term	Definition	Notes
4. 102	Cobol	index	A storage area or a register, the content of which refers to a particular element in a table.	
4. 103	Cobol	indexed organization	The permanent logical file structure in which each record is identified by the value of one or more keys within that record.	
4. 104	Cobol	inheritance (for classes)	A mechanism for using the interface and implementation of one or more classes as the basis for another class.	
4. 105	Cobol	inheritance (for interfaces)	A mechanism for using the specification of one or more interfaces as the basis for another interface.	
4. 106	Cobol	initial program	A program that is placed into the initial state every time the program is called.	
4. 107	Cobol	initial state	The state of a function, method, or program when it is first activated in a run unit.	
4. 108	Cobol	input mode	A mode of file processing in which records can only be read.	
4. 109	Cobol	instance object	A single instance of an object defined by a class and created by a factory object.	

Clause	Source	Term	Definition	Notes
4. 110	Cobol	interface (of an object)	The names of all the methods defined for the object, including inherited methods; for each of the methods: (1) the ordered list of its formal parameters and the description and passing technique associated with each, (2) any returned value and its description, (3) exceptions that may be raised.	
4. 111	Cobol	interface (the language construct)	A grouping of method prototypes.	
4. 112	Cobol	internal data	All data described in a source unit except external data and external file connectors.	
4. 113	Cobol	invocation; method invocation	See method invocation.	
4. 114	Cobol	key of reference	The key, either prime or alternate, currently being used to access records within an indexed file.	
4. 115	Cobol	letter	A basic letter or an extended letter.	
4. 116	Cobol	locale	A facility in the user's information technology environment that specifies language and cultural conventions.	
4. 117	Cobol	lock mode	The state of a file for which record locking is in effect that indicates whether record locking is manual or automatic.	
4. 118	Cobol	low-order end	The rightmost position of a string of characters or a string of bits.	

Clause	Source	Term	Definition	Notes
4. 119	Cobol	method	A procedural entity defined by a method definition within, or inherited by, a class definition as an allowable operation upon objects of that class.	
4. 120	Cobol	method data	Data declared in a method definition.	
4. 121	Cobol	method invocation; invocation	The request to execute a named method on a given object.	
4. 122	Cobol	method prototype	A source element that specifies the information needed for invocation of a method and for conformance checking.	
4. 123	Cobol	national character	A character in a national character set.	
4. 124	Cobol	national character position	The amount of physical storage required to store, or presentation space required to print or display, a single national character.	
4. 125	Cobol	national character set; national coded character set	See national coded character set.	
4. 126	Cobol	national coded character set; national character set	A coded character set that the implementor has designated for representation of data items described as usage national and for national literals.	
4. 127	Cobol	national data item	An elementary data item of class national or a national group item.	
4. 128	Cobol	native alphanumeric character set	The computer's alphanumeric coded character set.	

Clause	Source	Term	Definition	Notes
4. 129	Cobol	native arithmetic	A mode of arithmetic in which the techniques used in handling arithmetic are specified by the implementor.	
4. 130	Cobol	native character set	An implementor-defined character set, either alphanumeric or national or both, that is used for internal processing of a COBOL runtime module. The native character set is that referenced by the keyword NATIVE in the SPECIAL-NAMES paragraph.	
4. 131	Cobol	native collating sequence	An implementor-defined collating sequence, either an alphanumeric collating sequence or a national collating sequence, that is associated with the computer on which a runtime module is executed.	
4. 132	Cobol	native national coded character set	The computer's national coded character set.	
4. 133	Cobol	next record	The record that logically follows the current record of a file.	
4. 134	Cobol	null	The state of a pointer indicating that it contains no address, or the state of an object reference indicating that it contains no reference.	
4. 135	Cobol	numeric character (in the rules of COBOL)	A character that belongs to the following set of digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.	
4. 136	Cobol	object	A unit consisting of data and the methods that act upon that data.	

Clause	Source	Term	Definition	Notes
4. 137	Cobol	object data	Data defined: (1) in the factory definition, except for the data defined in its methods, (2) in the instance definition, except for the data defined in its methods.	
4. 138	Cobol	object property; property	A name that may be used to qualify an object reference to get a value from or pass a value to an object.	
4. 139	Cobol	object reference	An explicitly- or implicitly-defined data item that contains a reference to an object.	
4. 140	Cobol	open mode	The state of a file connector indicating input-output operations that are permitted for the associated file.	
4. 141	Cobol	optional file	A file declared as being not necessarily present each time the runtime module is executed.	
4. 142	Cobol	outermost program	A program, together with its contained programs, that is not contained in any other program.	
4. 143	Cobol	output file	A file that is opened in either the output mode or extend mode.	
4. 144	Cobol	output mode	A mode of file processing in which a file is created and records can only be added to the file.	
4. 145	Cobol	physical file	A physical collection of physical records.	
4. 146	Cobol	physical record; block	See block.	
4. 147	Cobol	previous record	The record that logically precedes the current record of a file.	

Clause	Source	Term	Definition	Notes
4. 148	Cobol	procedure	One or more successive paragraphs or sections in the procedure division.	
4. 149	Cobol	procedure branching statement	A statement that causes the explicit transfer of control to a statement other than the next executable statement in the sequence in which the statements are written.	
4. 150	Cobol	processor	The computing system, both hardware and software, used for compilation of source code or execution of run units, or both.	
4. 151	Cobol	program prototype definition	A definition that specifies the rules governing the class of the parameters expected to be received by a particular subprogram, and any other requirements needed to transfer control to and get control and return information from that subprogram.	
4. 152	Cobol	property; object property	See object property.	
4. 153	Cobol	random access	An access mode in which the value of a key data item identifies the logical record that is obtained from, deleted from, or placed into a relative or indexed file.	
4. 154	Cobol	record key	A data item within a record used to identify that record within an indexed file.	
4. 155	Cobol	record lock	A conceptual entity that is used to control concurrent access to a given record within a shared physical file.	

Clause	Source	Term	Definition	Notes
4. 156	Cobol	record locking	A facility for controlling concurrent access to records in a shared physical file.	
4. 157	Cobol	relative item	An item in a report whose position is specified relative to the previous item.	
4. 158	Cobol	relative key	A data item that contains a relative record number.	
4. 159	Cobol	relative organization	The permanent logical file structure in which each record's logical position is uniquely identified by a relative record number.	
4. 160	Cobol	relative record number	The ordinal number of a record in a file whose organization is relative.	
4. 161	Cobol	report	A printed output described in the report section and generated from those data descriptions.	
4. 162	Cobol	report writer	A comprehensive set of data clauses and statements that enable a print layout to be described according to its general appearance rather than through of a series of procedural steps.	
4. 163	Cobol	restricted pointer	A pointer data item that is restricted to data items of a specified type or to programs with the same signature as a specified program.	
4. 164	Cobol	run unit	One or more runtime modules that interact with one another and that function, at execution time, as an entity to provide problem solutions.	
4. 165	Cobol	runtime element	The executable unit resulting from compiling a source element.	

Clause	Source	Term	Definition	Notes
4. 166	Cobol	runtime module	The result of compiling a compilation unit.	
4. 167	Cobol	sequential access	An access method in which logical records are either placed into a file in the order of execution of the statements writing the records or obtained from a file in the sequence in which the records were written to the file.	
4. 168	Cobol	sequential organization	The permanent logical file structure in which a record is identified by a predecessor-successor relationship established when the record is placed into the file.	
4. 169	Cobol	sharing mode	The state of a file that indicates the mode of file sharing.	
4. 170	Cobol	source element	A source unit excluding any contained source units.	
4. 171	Cobol	source unit	A sequence of statements beginning with an identification division and finishing with an end marker or the end of the compilation group, including any contained source units.	
4. 172	Cobol	standard arithmetic	A mode of arithmetic in which the techniques used in handling arithmetic expressions, arithmetic statements, the SUM clause, and certain integer and numeric functions are specified in this Draft International Standard.	
4. 173	Cobol	standard intermediate data item	A temporary abstract decimal floating-point data item used to hold arithmetic operands when standard arithmetic is in effect.	

Clause	Source	Term	Definition	Notes
4. 174	Cobol	static data	Data that has its last-used state when a runtime element is re-entered.	
4. 175	Cobol	subclass	A class that inherits from another class. When two classes in an inheritance relationship are considered together, the subclass is the inheritor or inheriting class; the superclass is the inheritee or inherited class.	NOTE In the industry literature, the term derived class is also often used as an alternative to the term subclass. These terms are equivalent.
4. 176	Cobol	subject of the entry	The data item that is being defined by a data description entry.	
4. 177	Cobol	subscript	A number used to refer to a specific element of a table, or in the case of the subscript 'ALL', to all elements of a table.	
4. 178	Cobol	superclass	A class that is inherited by another class.	
4. 179	Cobol	surrogate pair	A coded character representation for a single abstract character of the UTF-16 format of the UCS where the representation consists of a sequence of two two-octet values. The first value of the pair is a high-surrogate and the second is a low-surrogate.	
4. 180	Cobol	type (for type declaration)	A template that contains all the characteristics of a data item and its subordinates.	
4. 181	Cobol	UCS; Universal Multiple-Octet Coded Character set.	See Universal Multiple-Octet Coded Character Set.	

Clause	Source	Term	Definition	Notes
4. 182	Cobol	Universal Multiple-Octet Coded Character Set; UCS.	The coded character set defined by ISO/IEC 10646-1 together with ISO/IEC 10646-2. This coded character set includes the characters used in writing most of the languages of the modern world.	
4. 183	Cobol	universal object reference	An object reference that is not restricted to a specific class or interface.	
4. 184	Cobol	unsuccessful execution	The attempted execution of a statement that does not result in the execution of all the operations specified by that statement.	
4. 185	Cobol	variable-occurrence data item	A table element that is repeated a variable number of times.	
4. 186	Cobol	zero-length item	An item whose minimum permitted length is zero and whose length at execution time is zero.	
3. 1	Extended BNF	sequence	An ordered list of zero or more items.	Note - This definition differs from the one in ISO 2382:1985.
3. 2	Extended BNF	subsequence	A sequence within a sequence.	
3. 3	Extended BNF	non-terminal symbol	A syntactic part of the language being defined.	
3. 4	Extended BNF	meta-identifier	The name of a non-terminal symbol.	
3. 5	Extended BNF	start symbol	A non-terminal symbol that is defined by one or more syntax rules but does not occur in any other syntax rule.	
3. 6	Extended BNF	sentence	A sequence of symbols that represents the start symbol.	

Clause	Source	Term	Definition	Notes
3. 7	Extended BNF	terminal symbol	A sequence of one or more characters forming an irreducible element of a language.	Note - In this International Standard a terminal symbol of Extended BNF is called a terminal-character, and a terminal symbol of a language being defined by a syntax is represented by a terminal-string.
An'x N	Ada	Access type	An access type has values that designate aliased objects. Access types correspond to ``pointer types'' or ``reference types'' in some other languages.	
An'x N	Ada	Aliased	An aliased view of an object is one that can be designated by an access value. Objects allocated by allocators are aliased. Objects can also be explicitly declared as aliased with the reserved word aliased. The Access attribute can be used to create an a	
An'x N	Ada	Array type	An array type is a composite type whose components are all of the same type. Components are selected by indexing.	
An'x N	Ada	Character type	A character type is an enumeration type whose values include characters.	
An'x N	Ada	Class	A class is a set of types that is closed under derivation, which means that if a given type is in the class, then all types derived from that type are also in the class. The set of types of a class share common properties, such as their primitive operati	
An'x N	Ada	Compilation unit	The text of a program can be submitted to the compiler in one or more compilations. Each compilation is a succession of compilation_units. A compilation_unit contains either the declaration, the body, or a renaming of a program unit.	
An'x N	Ada	Composite type	A composite type has components.	
An'x N	Ada	Construct	A construct is a piece of text (explicit or implicit) that is an instance of a syntactic category defined under ``Syntax.''	

Clause	Source	Term	Definition	Notes
An'x N	Ada	Controlled type	A controlled type supports user-defined assignment and finalization. Objects are always finalized before being destroyed.	
An'x N	Ada	Declaration	A declaration is a language construct that associates a name with (a view of) an entity. A declaration may appear explicitly in the program text (an explicit declaration), or may be supposed to occur at a given place in the text as a consequence of the s	
An'x N	Ada	Definition	All declarations contain a definition for a view of an entity. A view consists of an identification of the entity (the entity of the view), plus view-specific characteristics that affect the use of the entity through that view (such as mode of access to	
An'x N	Ada	Derived type	A derived type is a type defined in terms of another type, which is the parent type of the derived type. Each class containing the parent type also contains the derived type. The derived type inherits properties such as components and primitive operation	
An'x N	Ada	Discrete type	A discrete type is either an integer type or an enumeration type. Discrete types may be used, for example, in case_statements and as array indices.	
An'x N	Ada	Discriminant	A discriminant is a parameter of a composite type. It can control, for example, the bounds of a component of the type if that type is an array type. A discriminant of a task type can be used to pass data to a task of the type upon creation.	
An'x N	Ada	Elementary type	An elementary type does not have components.	
An'x N	Ada	Enumeration type	An enumeration type is defined by an enumeration of its values, which may be named by identifiers or character literals.	

Clause	Source	Term	Definition	Notes
An'x N	Ada	Exception	An exception represents a kind of exceptional situation; an occurrence of such a situation (at run time) is called an exception occurrence. To raise an exception is to abandon normal program execution so as to draw attention to the fact that the correspo	
An'x N	Ada	Execution	The process by which a construct achieves its run-time effect is called execution. Execution of a declaration is also called elaboration. Execution of an expression is also called evaluation.	
An'x N	Ada	Generic unit	A generic unit is a template for a (nongeneric) program unit; the template can be parameterized by objects, types, subprograms, and packages. An instance of a generic unit is created by a generic_instantiation. The rules of the language are enforced when	
An'x N	Ada	Integer type	Integer types comprise the signed integer types and the modular types. A signed integer type has a base range that includes both positive and negative numbers, and has operations that may raise an exception when the result is outside the base range. A mo	
An'x N	Ada	Library unit	A library unit is a separately compiled program unit, and is always a package, subprogram, or generic unit. Library units may have other (logically nested) library units as children, and may have other program units physically nested within them. A root	
An'x N	Ada	Limited type	A limited type is (a view of) a type for which the assignment operation is not allowed. A nonlimited type is a (view of a) type for which the assignment operation is allowed.	
An'x N	Ada	Object	An object is either a constant or a variable. An object contains a value. An object is created by an object_declaration or by an allocator. A formal parameter is (a view of) an object. A subcomponent of an object is an object.	

Clause	Source	Term	Definition	Notes
An'x N	Ada	Package	Packages are program units that allow the specification of groups of logically related entities. Typically, a package contains the declaration of a type (often a private type or private extension) along with the declarations of primitive subprograms of t	
An'x N	Ada	Partition	A partition is a part of a program. Each partition consists of a set of library units. Each partition may run in a separate address space, possibly on a separate computer. A program may contain just one partition. A distributed program typically contains	
An'x N	Ada	Pragma	A pragma is a compiler directive. There are language-defined pragmas that give instructions for optimization, listing control, etc. An implementation may support additional (implementation-defined) pragmas.	
An'x N	Ada	Primitive operations	The primitive operations of a type are the operations (such as subprograms) declared together with the type declaration. They are inherited by other types in the same class of types. For a tagged type, the primitive subprograms are dispatching subprogram	
An'x N	Ada	Private extension	A private extension is like a record extension, except that the components of the extension part are hidden from its clients.	
An'x N	Ada	Private type	A private type is a partial view of a type whose full view is hidden from its clients.	
An'x N	Ada	Program	A program is a set of partitions, each of which may execute in a separate address space, possibly on a separate computer. A partition consists of a set of library units.	
An'x N	Ada	Program unit	A program unit is either a package, a task unit, a protected unit, a protected entry, a generic unit, or an explicitly declared subprogram other than an enumeration literal. Certain kinds of program units can be separately compiled. Alternatively, they c	

Clause	Source	Term	Definition	Notes
An'x N	Ada	Protected type	A protected type is a composite type whose components are protected from concurrent access by multiple tasks.	
An'x N	Ada	Real type	A real type has values that are approximations of the real numbers. Floating point and fixed point types are real types.	
An'x N	Ada	Record extension	A record extension is a type that extends another type by adding additional components.	
An'x N	Ada	Record type	A record type is a composite type consisting of zero or more named components, possibly of different types.	
An'x N	Ada	Scalar type	A scalar type is either a discrete type or a real type.	
An'x N	Ada	Subtype	A subtype is a type together with a constraint, which constrains the values of the subtype to satisfy a certain condition. The values of a subtype are a subset of the values of its type.	
An'x N	Ada	Tagged type	The objects of a tagged type have a run-time type tag, which indicates the specific type with which the object was originally created. An operand of a class-wide tagged type can be used in a dispatching call; the tag indicates which subprogram body to in	
An'x N	Ada	Task type	A task type is a composite type whose values are tasks, which are active entities that may execute concurrently with other tasks. The top-level task of a partition is called the environment task.	
An'x N	Ada	Type	Each object has a type. A type has an associated set of values, and a set of primitive operations which implement the fundamental aspects of its semantics. Types are grouped into classes. The types of a given class share a set of primitive operations. Cl	
An'x N	Ada	View	(See Definition.)	
An'x A	ASIS	Ancestor	Ancestors of a library unit are itself, its parent, its parent's parent, and so on. (Standard is an ancestor of every library unit).	

Clause	Source	Term	Definition	Notes
An'x A	ASIS	ASIS application	Any programming system or any set of software components making use of ASIS queries to obtain information about any set of Ada components.	
An'x A	ASIS	ASIS implementation	All the hardware and software that implement the ASIS specification for a given Ada implementation and that provide the functionality required by the ASIS specification.	
An'x A	ASIS	ASIS queries	Those subprogram interfaces (and only those) defined in the ASIS standard; these are supported by types, subtypes, and exceptions also defined in the ASIS standard. Thus, ASIS queries and supporting entities are together the ASIS interface. The followi	(Note that semantic queries are generally named "Corresponding_..." or "Implicit_..." in the ASIS specification.)
An'x A	ASIS	Closure	A term commonly used instead of needed units.	
An'x A	ASIS	Compilation unit	“The term compilation unit is used to refer to a compilation_unit. When the meaning is clear from context, the term is also used to refer to the library_item of a compilation_unit or to the proper_body of a subunit”. [ISO/IEC 8652:1995, 10.1.1(9)]. AS	Note, that the term "compilation unit" can refer to either syntactical category "compilation_unit" or to the library_item of a compilation_unit or to the proper_body of a subunit (that is, the compilation_unit without the context_clause and the separate (
An'x A	ASIS	Compilation_Unit [type]	An ASIS private type for which values denote an Ada compilation unit or configuration pragma from the environment denoted by some open ASIS context. A non-nil value of the Compilation_Unit type also contains information about some physical object from the	
An'x A	ASIS	Container	Logical collection of ASIS compilation units. For example, some container can hold compilation units which include Ada predefined types, another container can hold implementation-defined packages. Containers provide the implementation-defined way of gr	

Clause	Source	Term	Definition	Notes
An'x A	ASIS	Container [type]	An ASIS private type for which values denote a set of compilation units being a subset of the set of compilation units making up a context.	
An'x A	ASIS	Context	Defines a set of compilation units and configuration pragmas processed by an ASIS application. ASIS provides any information from a context by treating this set as if its elements make up an environment declarative part by modeling some view (most likel	
An'x A	ASIS	Context [type]	An ASIS private type for which values denote a set of compilation units considered by ASIS as making up an Ada environment declarative part from which to provide information.	
An'x A	ASIS	Dependent	Dependents of a compilation unit are all the compilation units that depend semantically on it, either directly or indirectly. A is a dependent of B, if B is a supporter of A.	
An'x A	ASIS	Descendants	Descendants of a library unit relation are the inverse of the ancestor relation.	
An'x A	ASIS	Element	A common abstraction used by ASIS to denote the syntax components (both explicit and implicit) of ASIS compilation units. The term Element is also used as the synonym for "the value of the ASIS Element type". See also "Explicit element" and "Implicit e	
An'x A	ASIS	Element [type]	An ASIS private type, whose values represent the syntax components (both explicit and implicit) of ASIS compilation units.	
An'x A	ASIS	Environment	"Each compilation unit submitted to the compiler is compiled in the context of an environment declarative_part (or simply environment), which is a conceptual declarative_part that forms the outermost declarative region of the context of any compilation	Note that the mechanisms for creating an environment and for adding and replacing compilation units within an environment are implementation-defined.
An'x A	ASIS	Explicit element	An ASIS Element, representing a language construct that appears explicitly in the program text for the compilation unit (e.g., an explicit declaration).	

Clause	Source	Term	Definition	Notes
An'x A	ASIS	Extension	Non-standard facilities (other library units, non-standard children of standard ASIS library units, subprograms, etc.) which provide additional information from ASIS types, or modify the behavior of otherwise standard ASIS facilities to provide alternat	
An'x A	ASIS	Family	The family of a given unit is defined as the set of compilation units that comprise the given unit's declaration, body, descendants, and subunits (and subunits of subunits and descendants, etc.).	
An'x A	ASIS	Id	A way of identifying a particular element, from a particular compilation unit, from a particular context.	
An'x A	ASIS	Id [type]	An ASIS private type implementing the Id abstraction. The values of this type can be written to files. These values can be read back from files and converted into values of the Element type with the use of a suitable open context.	
An'x A	ASIS	Implementor	A company, institution, or other group (such as a vendor) who develops an ASIS implementation; thus an ASIS implementor. There are also Ada implementors, who provide Ada compilation systems; and there are ASIS-based tool (or, ASIS Application) implemen	
An'x A	ASIS	Implicit element	An ASIS Element, representing a language construct that does not exist in the program text for the compilation unit, but could occur at a given place in the program text as a consequence of the semantics of another construct, (e.g., an implicit declarat	
An'x A	ASIS	Line	The logical representation of a line of text from the source code of the external representation of a compilation unit.	

Clause	Source	Term	Definition	Notes
An'x A	ASIS	Line [type]	An ASIS private type for the ASIS Line abstraction. The values of the Line type represent the lines of text from the source code of the external representation of compilation units.	
An'x A	ASIS	Needed Units	The needed units of a given compilation unit is a set of compilation units ultimately needed by the given compilation unit to make up or to be included in a completed partition.	
An'x A	ASIS	Optional functionality	The subset of ASIS facilities that are explicitly identified in the ASIS standard as optional which may legitimately be omitted from a Basic Conforming ASIS implementation, but shall be included in any Fully Conforming ASIS implementation, unless stated	
An'x A	ASIS	Queries.	See ASIS queries.	
An'x A	ASIS	Relation (between ASIS Compilation Units)	Semantic relationships between compilation units (as discussed in chapter 10 of ISO/IEC 8652:1995). The Relation_Kindstype enumerates the kinds of relations that can exist between compilation units. See also Dependent, Extended Family, and Supporter.	
An'x A	ASIS	Required functionality	The subset of ASIS facilities which are not explicitly identified in the ASIS standard as optional which shall be included in a Basic or Fully Conforming ASIS implementation, unless stated otherwise in the ASIS specification.	
An'x A	ASIS	Semantic queries.	See ASIS queries.	
An'x A	ASIS	Structural queries.	See ASIS queries.	
An'x A	ASIS	Supporter	Supporters of a compilation unit are units on which it semantically depends, either directly or indirectly. B is a supporter of A, if A is a dependent of B.	

Clause	Source	Term	Definition	Notes
4.1. 1	ACATS	consensus	general agreement, characterized by the absence of sustained opposition to substantial issues by any important part of the concerned interests and by a process that involves seeking to take into account the views of all parties concerned and to reconcile	NOTE - Consensus need not imply unanimity. [ISO/IEC Guide 2, 1.7]
4.2. 1	ACATS	fitness for purpose	ability of a product, process or service to serve a defined purpose under specific conditions	[ISO/IEC Guide 2, 2.1]
4.3. 1	ACATS	document	any medium with information recorded on or in it	[ISO/IEC Guide 2, 3.1]
4.3. 2	ACATS	normative document	document that provides rules, guidelines or characteristics for activities or their results	NOTES -1 - The term “normative document” is a generic term that covers such documents as standards, technical specifications, codes of practice and regulations.2 - The terms for different kinds of normative documents are defined considering the document a
4.3. 3	ACATS	standard	document, established by consensus and approved by a recognized body, that provides, 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 cont	[ISO/IEC Guide 2, 3.2]
4.3. 4	ACATS	International Standard	standard that is adopted by an international standardizing/standards organization and made available to the public	[ISO/IEC Guide 2, 3.2.1.1]
4.3. 5	ACATS	technical specification	document that prescribes technical requirements to be fulfilled by a product, process or service	NOTES = 1 - A technical specification would indicate, whenever appropriate, the procedure(s) by means of which it may be determined whether the requirements given are fulfilled.2 - A technical specification may be a standard, a part of a standard or indep
4.3. 6	ACATS	regulation	document providing binding legislative rules, that is adopted by an authority	[ISO/IEC Guide 2, 3.6]

Clause	Source	Term	Definition	Notes
4.4. 1	ACATS	body	legal or administrative entity that has specific tasks and composition	NOTE - Examples of bodies are organizations, authorities, companies and foundations. [ISO/IEC Guide 2, 4.1]
4.4. 2	ACATS	organization	body that is based on the membership of other bodies or individuals and has an established constitution and its own administration	[ISO/IEC Guide 2, 4.2]
4.4. 3	ACATS	international standardizing organization	standardizing organization whose membership is open to the relevant national body from every country	[ISO/IEC Guide 2, 4.3.2]
4.4. 4	ACATS	authority	body that has legal powers and rights	NOTE - An authority can be regional, national or local. [ISO/IEC Guide 2, 4.5]
4.5. 1	ACATS	testing standard	standard that is concerned with test methods, sometimes supplemented with other provisions related to testing, such as sampling, use of statistical methods, sequence of tests	[ISO/IEC Guide 2, 5.3]
4.5. 2	ACATS	product standard	standard that specifies requirements to be fulfilled by a product or a group of products, to establish its fitness for purpose	[ISO/IEC Guide 2, 5.4]
4.6. 1	ACATS	provision	expression in the content of a normative document, that takes the form of a statement, an instruction, a recommendation or a requirement	NOTE - These types of provision are distinguished by the form of wording they employ; e.g. instructions are expressed in the imperative mood, recommendations by the use of the auxiliary “should” and requirements by the use of the auxiliary “shall.” [ISO/I
4.6. 2	ACATS	statement	provision that conveys information	[ISO/IEC Guide 2, 7.2]
4.6. 3	ACATS	instruction	provision that conveys an action to be performed	[ISO/IEC Guide 2, 7.3]
4.6. 4	ACATS	recommendation	provision that conveys advice or guidance	[ISO/IEC Guide 2, 7.4]
4.6. 5	ACATS	requirement	provision that conveys criteria to be fulfilled	[ISO/IEC Guide 2, 7.5]
4.6. 6	ACATS	exclusive requirement	requirement of a normative document that must necessarily be fulfilled in order to comply with that document	NOTE - The term “mandatory requirement” should be used to mean only a requirement made compulsory by law or regulation. [ISO/IEC Guide 2, 7.5.1]

Clause	Source	Term	Definition	Notes
4.6. 7	ACATS	optional requirement	requirement of a normative document that must be fulfilled in order to comply with a particular option permitted by that document	NOTE - An optional requirement may be either (a) one of two or more alternative requirements; or(b) an additional requirement that must be fulfilled only if applicable and that may otherwise be disregarded. [ISO/IEC Guide 2, 7.5.2]
4.6. 8	ACATS	deemed-to-satisfy provision	provision that indicates one or more means of compliance with a requirement of a normative document	[ISO/IEC Guide 2, 7.6]
4.6. 9	ACATS	descriptive provision	provision for fitness for purpose that concerns the characteristics of a product, process or service	NOTE - A descriptive provision usually conveys design, constructional details, etc. with dimensions and material composition. [ISO/IEC Guide 2, 7.7]
4.7. 1	ACATS	conformity	fulfillment by a product, process or service of specified requirements	[ISO/IEC Guide 2, 12.1]
4.7. 2	ACATS	conformity assessment	any activity concerned with determining directly or indirectly that relevant requirements are fulfilled	NOTE - Typical examples of conformity assessment activities are sampling, testing and inspection; evaluation, verification and assurance of conformity (supplier's declaration, certification); registration, accreditation and approval as well as their combi
4.7. 3	ACATS	conformity assessment body	body that conducts conformity assessment	[ISO/IEC Guide 2, 12.3]
4.7. 4	ACATS	conformity assessment system	system that has its own rules of procedure and management for carrying out conformity assessment	NOTES - 1 - Conformity assessment systems may be operated at, for example, national, regional or international level.2 - Typical examples of conformity assessment systems are testing systems, inspection systems, and certification systems. [ISO/IEC Guide 2

Clause	Source	Term	Definition	Notes
4.7. 5	ACATS	conformity assessment scheme	conformity assessment system as related to specified products, processes or services to which the same particular standards and rules, and the same procedure, apply	NOTE - The term “program” is used in some countries to cover the same concept as “scheme.” [ISO/IEC Guide 2, 12.5]
4.7. 6	ACATS	access to a conformity assessment system	opportunity for an applicant to obtain conformity assessment under the rules of the system	[ISO/IEC Guide 2, 12.6]
4.7. 7	ACATS	participant in a conformity assessment system	conformity assessment body that operates under the rules of the system without having the opportunity to take part in the management of the system	[ISO/IEC Guide 2, 12.7]
4.7. 8	ACATS	member of a conformity assessment system	conformity assessment body that operates under the rules of the system and has the opportunity to take part in the management of the system	[ISO/IEC Guide 2, 12.8]
4.7. 9	ACATS	third party	person or body that is recognized as being independent of the parties involved, as concerns the issue in question	NOTE - Parties involved are usually supplier (“first party”) and purchaser (“second party”) interests. [ISO/IEC Guide 2, 12.9]
4.7. 10	ACATS	registration	procedure by which a body indicates relevant characteristics of a product, process or service, or particulars of a body or person, in an appropriate, publicly available list	[ISO/IEC Guide 2, 12.10]
4.7. 11	ACATS	accreditation	procedure by which an authoritative body gives formal recognition that a body or person is competent to carry out specific tasks	[ISO/IEC Guide 2, 12.11]
4.7. 12	ACATS	reciprocity	bilateral relationship where both parties have the same rights and obligations towards each other	[ISO/IEC Guide 2, 12.12]
4.7. 13	ACATS	equal treatment	treatment accorded to products, processes or services originating in other countries that is no less favorable than that accorded to like products, processes or services of national origin, in a comparable situation	[ISO/IEC Guide 2, 12.13]

Clause	Source	Term	Definition	Notes
4.8. 1	ACATS	test	technical operation that consists of the determination of one or more characteristics of a given product, process or service according to a specified procedure	[ISO/IEC Guide 2, 13.1]
4.8. 2	ACATS	testing	action of carrying out one or more tests	[ISO/IEC Guide 2, 13.1.1]
4.8. 3	ACATS	test method	specified technical procedure for performing a test	[ISO/IEC Guide 2, 13.2]
4.8. 4	ACATS	test report	document that presents test results and other information relevant to a test	[ISO/IEC Guide 2, 13.3]
4.8. 5	ACATS	laboratory	body that calibrates and/or tests	[ISO/IEC Guide 25, 3.1]
4.8. 6	ACATS	testing laboratory	laboratory that performs tests	NOTE - The term “testing laboratory” can be used in the sense of a legal entity, a technical entity or both. [ISO/IEC Guide 2, 13.4]
4.9. 1	ACATS	conformity evaluation	systematic examination of the extent to which a product, process or service fulfills specified requirements	[ISO/IEC Guide 2, 14.1]
4.9. 2	ACATS	inspection	conformity evaluation by observation and judgment accompanied as appropriate by measurement, testing or gauging	[ISO/IEC Guide 2, 14.2]
4.9. 3	ACATS	inspection body	body that performs inspection	[ISO/IEC Guide 2, 14.3]
4.9. 4	ACATS	verification	confirmation by examination and provision of evidence that specified requirements have been met	[ISO/IEC Guide 25, 3.8]
4.9. 5	ACATS	conformity testing	conformity evaluation by means of testing	[ISO/IEC Guide 2, 14.4]
4.9. 6	ACATS	type testing	conformity testing on the basis of one or more specimens of a product representative of the production	[ISO/IEC Guide 2, 14.5]
4.10. 1	ACATS	assurance of conformity	activity resulting in a statement giving confidence that a product, process or service fulfills specified requirements	NOTE - For a product, the statement may be in the form of a document, a label or other equivalent means. It may also be printed in or applied on a communication, a catalogue, an invoice, a user instructions manual, etc. relating to the product. [ISO/IEC

Clause	Source	Term	Definition	Notes
4.10. 2	ACATS	supplier’s declaration	procedure by which a supplier gives written assurance that a product, process or service conforms to specified requirements	NOTE - In order to avoid any confusion, the expression “self-certification” should not be used. [ISO/IEC Guide 2, 15.1.1]
4.10. 3	ACATS	certification	procedure by which a third party gives written assurance that a product, process or service conforms to specified requirements	[ISO/IEC Guide 2, 15.1.2]
4.10. 4	ACATS	certification body	body that conducts certification	NOTE - A certification body may operate its own testing and inspection activities or oversee these activities carried out on its behalf by other bodies. [ISO/IEC Guide 2, 15.2]
4.10. 5	ACATS	certificate of conformity	a document issued under the procedures of a third party certification system and attesting that a product or service is in conformity with specific standards or other technical specifications	[ISO/IEC Guide 23, 3.2; ISO/IEC Guide 2, 15.5]
4.10. 6	ACATS	mark of conformity	a legally registered certification mark applied by or issued under the procedures of a third party certification system for a product or service which is in conformity with specific standards or other technical specifications	[ISO/IEC Guide 23, 3.1; superseding ISO/IEC Guide 2, 15.6]
4.11. 1	ACATS	approval	permission for a product, process or service to be marketed or used for stated purposes or under stated conditions	[ISO/IEC Guide 2, 16.1]
4.11. 2	ACATS	type approval	approval based on type testing	[ISO/IEC Guide 2, 16.1.1]
4.11. 3	ACATS	recognition arrangement	agreement that is based on the acceptance by one party of results, presented by another party, from the implementation of one or more designated functional elements of a conformity assessment system	NOTES - 1 - Typical examples of recognition arrangements are testing arrangements, inspection arrangements and certification arrangements.2 - Recognition arrangements may be established at, for example, national, regional or international level.3 - An agr
4.11. 4	ACATS	multilateral arrangement	recognition arrangement that covers the acceptance of each other’s results by more than two parties	[ISO/IEC Guide 2, 16.5]

Clause	Source	Term	Definition	Notes
4.12. 1	ACATS	accreditation system	system that has its own rules of procedure and management for carrying out accreditation	NOTE - Accreditation of conformity assessment bodies is normally awarded following successful assessment and is followed by appropriate surveillance.[ISO/IEC Guide 2, 17.1]
4.12. 2	ACATS	accreditation body	body that conducts and administers an accreditation system and grants accreditation	[ISO/IEC Guide 2, 17.2]
4.12. 3	ACATS	accredited body	body to which accreditation has been granted	[ISO/IEC Guide 2, 17.3]
4.12. 4	ACATS	accreditation criteria	set of requirements that is used by an accreditation body, to be fulfilled by a conformity assessment body in order to be accredited	[ISO/IEC Guide 2, 17.4]
4.13. 1	ACATS	configuration	host and target computers, any operating system(s) and software used to operate a processor	[ISO TR 9547, 2.1]
4.13. 2	ACATS	extension	a facility in the implemented language that is not given in the language standard but that does not cause any ambiguity or contradiction when added to the language standard (although, in some languages, it may serve to lift a restriction)	[ISO TR 9547, 2.2]
4.13. 3	ACATS	implementation defined	dependent on the processor but required by the language standard to be defined and documented by the implementer	[ISO TR 9547, 2.3]
4.13. 4	ACATS	processor	a compiler, translator or interpreter working in combination with a configuration	[ISO TR 9547, 2.4]
4.13. 5	ACATS	test program	a sequence of characters intended to be submitted to a processor in order to determine whether or not this processor exhibits a specific instance of a certain property	[ISO TR 9547, 2.7]
4.13. 6	ACATS	test suite	a reference set of test programs that is designed to assess conformity of a processor with a language standard	[ISO TR 9547, 2.9]
4.13. 7	ACATS	test tools	any additional means that can improve the efficiency, the reliability and the ease of use of the different phases of testing (e.g. implementation of the test suite, ensuring integrity, processing of the test suite, collecting test results, analysis of tes	[ISO TR 9547, 2.10]

Clause	Source	Term	Definition	Notes
4.13. 8	ACATS	required documents	the set of documents required by the programming language standard	[ISO TR 9547, 2.11]
4.13. 9	ACATS	subset	a subset S of a programming language L is a programming language such that every program in S is also a program in L and has the same meaning in S as it has in L	[ISO TR 9547, 2.12]
4.14. 1	ACATS	quality manual	a document stating the quality policy, quality system and quality practices of an organization	NOTES - 1 - The quality manual may call up other documentation relating to the organization's quality arrangements.2 - The quality manual may be a distinct part of other documentation. [ISO/IEC Guide 25, 3.10]
4.14. 2	ACATS	Core Language	the provisions of clauses 1-13 and Annexes A, B, and J of ISO/IEC 8652	NOTE - Conformity to the Core Language is required by any Ada language processor. [ISO/IEC 8652, 1.1.2]
4.14. 3	ACATS	Specialized Needs Annexes	Annexes C through H of ISO/IEC 8652	NOTE - An Ada language processor may conform to some or none of these Annexes. [ISO/IEC 8652, 1.1.2]
4.15. 1	ACATS	Ada Conformity Assessment Process	the process by which conformity of Ada language processors to the language standard, ISO/IEC 8652, is assessed.	
4.15. 2	ACATS	Ada Conformity Assessment Procedure (ACAP)	detailed provisions, instructions, requirements and descriptions of processes regarding all aspects of the Ada Conformity Assessment Process collected in a document.	
4.15. 3	ACATS	Ada Conformity Assessment Laboratory (ACAL)	an independent testing laboratory conducting conformity assessment tests in accordance with this International Standard.	
4.15. 4	ACATS	Ada Conformity Assessment Authority (ACAA)	an organization that ensures world-wide commonality of the Ada Conformity Assessment Process.	
4.15. 5	ACATS	Ada Conformity Assessment Test Suite (ACATS)	the test suite used in the Ada Conformity Assessment Process .	

Clause	Source	Term	Definition	Notes
4.15. 6	ACATS	certification by derivation	registration of conforming processors obtained by adaptive and perfective maintenance from a processor for which conformity of the processor was successfully assessed by witness-testing on the same or a closely related configuration.	
4.15. 7	ACATS	certification by extension	registration of a conforming processor on configurations closely related to the configuration on which conformity of the processor was successfully assessed by witness-testing.	
4.15. 8	ACATS	Declaration of Conformity	a statement, signed by an authorized officer of the manufacturer of an Ada language processor, asserting that the manufacturer has no knowledge of an intentional deviation of the Ada language processor from the Ada language standard.	
4.15. 9	ACATS	client	an organization that obtains conformity assessment services from an ACAL .	
4.15. 10	ACATS	manufacturer	an organization responsible for the production and maintenance of a language processor.	
4.15. 11	ACATS	self-testing	the processing of an appropriately customized version of the ACATS, but not under the observation of an ACAL.	
4.15. 12	ACATS	test issue	any disagreement between an ACAL and its client over the conduct of the conformity assessment and, in particular, any disagreement over the fitness for purpose of any test in the ACATS.	
4.15. 13	ACATS	witness testing	the processing of an appropriately customized version of the ACATS under the observation of an ACAL.	