AD-A282 324

Report No. NAWCADWAR-94109-70

POSIX Delta Document for the Next-Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline (Version 4)

Operating Systems Standards Working Group (OSSWG)
Compiled by F. Prindle (NAWC-AD Warminster)

1 June 1994

Approved for Public Release; Distribution is Unlimited

Prepared for
Space and Naval Warfare Systems Command (SPAWAR 331-2)
Next Generation Computer Resources (NGCR) Program Office
2451 Crystal Drive
Arlington, VA 22245

NOTICES

REPORT NUMBERING SYSTEM — The numbering of technical project reports issued by the Naval Air Warfare Center, Aircraft Division, Warminster is arranged for specific identification purposes. Each number consists of the Center acronym, the calendar year in which the number was assigned, the sequence number of the report within the specific calendar year, and the official 2-digit correspondence code of the Functional Department responsible for the report. For example: Report No. NAWCADWAR-92001-60 indicates the first Center report for the year 1992 and prepared by the Air Vehicle and Crew Systems Technology Department. The numerical codes are as follows:

CODE	OFFICE OR DEPARTMENT
00	Commanding Officer, NAWCADWAR
01	Technical Director, NAWCADWAR
05	Computer Department
10	AntiSubmarine Warfare Systems Department
20	Tactical Air Systems Department
30	Warfare Systems Analysis Department
50	Mission Avionics Technology Department
60	Air Vehicle & Crew Systems Technology Department
70	Systems & Software Technology Department
80	Engineering Support Group
90	Test & Evaluation Group

Reviewed By: [signature] Date: 6/13/94
Branch Head

Reviewed By: [signature] acting Date: 13 June 94
Division Head

Reviewed By: [signature] Date: 6/15/94
Director/Deputy Director

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 June 1994	Interim

4. TITLE AND SUBTITLE
POSIX Delta Document for the Next-Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline (Version 4)

5. FUNDING NUMBERS

6. AUTHOR(S)
Operating Systems Standards Working Group (OSSWG)
Compiled by F. Prindle (NAWC-AD Warminster)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Air Warfare Center
Aircraft Division
Warminster, PA 18974-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NAWCADWAR-94109-70

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Command (SPAWAR 331-2)
Next Generation Computer Resources (NGCR) Program Office
2451 Crystal Drive
Arlington, VA 22245

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution is Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

The objective of the Next-Generation Computer Resources (NGCR) Program is to standardize computer and computer component interfaces for the Navy's next generation of mission-critical computing systems. The NGCR Operating Systems Standards Working Group (OSSWG) will either select a set of interface standards from commercial sources or jointly develop such standards with industry. Previously, the OSSWG established operating systems interface requirements for these standards. In this document, the OSSWG evaluates how effectively its established requirements are met by the Portable Operating System Interface (POSIX) standards, which have been selected as the baseline for the NGCR Program.

14. SUBJECT TERMS
Computer Operating Systems
Operating System Interface Standards

15. NUMBER OF PAGES
108

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

PREFACE

This report was funded under NAWC-AD Project No. 5109547, "Next Generation Computer Resources (NGCR)." The sponsoring activity is the Space and Naval Warfare Systems Command (SPAWAR), through the work of the Operating Systems Standards Working Group (OSSWG). The OSSWG management structure is as follows:

NGCR Program Manager, H. Mendenhall (SPAWAR-231)

NGCR OSSWG Cochairman, CDR M. Aaby (SPAWAR-331)

NGCR OSSWG Cochairman, J. Oblinger (NUWC)

Approach Subgroup Chairman, S. Howell (NSWC)

Standards Subgroup Chairman, F. Prindle (NAWC-AD)

Handbook Subgroup Chairman, D. Juttelstad (NUWC)

Table of Contents

Table of Contents (Cont'd)

Section **Page**

Table of Contents (Cont'd)

Table of Contents (Cont'd)

POSIX DELTA DOCUMENT FOR THE NEXT-GENERATION COMPUTER RESOURCES (NGCR) OPERATING SYSTEMS INTERFACE STANDARD BASELINE (VERSION 4)

1. INTRODUCTION

The objective of the Next-Generation Computer Resources (NGCR) Program is to standardize Navy mission-critical computer interfaces and computer component interfaces. With these standardized interfaces, industry will be better able to provide computing resources that meet Navy needs. The interface standards are to be widely available (i.e., non-proprietary) and, if possible, widely used within industry.

The NGCR Operating Systems Standards (OSS) is one of the sets of standards essential to the timely and cost effective acquisition of most of the next generation of mission-critical computing systems for the Navy. NGCR OSS assists the Navy in efficiently providing a wide range of performance, compatible computing services, and functionality levels.

The primary objective of the NGCR Operating Systems Standards Working Group (OSSWG) will be the selection, from commercial standards, of a set of interface standards for a family of distributed operating systems applicable to a complete spectrum of Navy combatant use and other mission-critical use. If these standards are not available or adequate, a standard will be developed in conjunction with industry.

1.1 SCOPE

The scope of this document includes the NGCR OSSWG Operational Concept Document (NUWC Technical Document 10168, February 1993) and all available documents, draft and final, from the family of the Portable Operating System Interfaces (POSIX) standards, which have been selected as the NGCR baseline. In addition, the documents from the IEEE working groups 1201, 1224. 1238, 1326, 1327, 1328, 1351, and 1353 were examined.

1.2 PURPOSE

The purpose of this document is to evaluate how effectively each Operating System Interface (OSIF) requirement, as defined by the Operational Concept Document, is addressed by the POSIX standards. By evaluating each OSIF requirement, the OSSWG will be able to determine as to how well the POSIX standards currently meet the Navy's needs.

The findings of this document will form a basis for identifying enhancements to POSIX. Comparing the POSIX standards and OSIF requirements can lead to one of several findings:

Requirement is fulfilled by POSIX,
Requirement is unnecessary and can be discarded,
Requirement is fulfilled by SAFENET,
Requirement was previously considered and discarded by POSIX,
Requirement is nice to have, but not really needed or worth working toward,
Requirement is "too far out" and it would be premature to standardize at this time,
Requirement is a must ("got to have") and must be included even if POSIX does not include it,
POSIX includes this useful feature but it is not a requirement.

From the list of requirements being pursued, an approach to take them into POSIX must be determined, explaining the concepts, rationale, and interfaces required.

If a necessary requirement conflicts with POSIX, then the OSSWG will develop a strategy for meeting this requirement. This document will eventually become a primary input into a Military Handbook for an OSIF. All requirements not fulfilled by POSIX standards or some other open standard will be addressed in the Military Handbook.

1.3 TERMINOLOGY

Precise and consistent use of terms has been attempted throughout the document. The following verb phrases are used in all NGCR documents to indicate where and to what degree individual constraints apply:

"SHALL PROVIDE" indicates a requirement for the operating system interface to provide interface(s) with prescribed capabilities.

"SHALL SUPPORT" indicates a requirement for the operating system interface to provide interface(s) with prescribed capabilities or for the operating system interface definers to demonstrate that the capability can be constructed from operating system interfaces.

1.4 DOCUMENT ORGANIZATION

This document was originally organized to reflect the evolutionary analysis process utilized by the OSSWG to determine, for each OSSWG requirement, the extent to which POSIX fulfilled that requirement, the overall importance of the requirement being fulfilled by standard interfaces, and the OSSWG approach to defining standard interfaces to fulfill all critical requirements. However, this organization was considered awkward, at best, for document maintenance and reader comprehension. Since the original analysis process is long since completed, it is no longer necessary for the document to retain this structure. Therefore, starting with version 4, the document has been reorganized more along the lines of a reference guide. The historical information on the analysis process can always be found in earlier document versions.

The current structure of the document is centered around section 3, where each operating system interface requirement from the Operational Concept Document (OCD) is presented, grouped into the same service classes and in the same order as defined in the OCD. For a requirement which is completely fulfilled by POSIX, this section indicates which POSIX interfaces fulfill the requirement, and provides an explanation of how this is accomplished where it isn't completely obvious. For a requirement which is either partially or totally unfulfilled by POSIX, this section describes: the extent of the delta (partial or no POSIX coverage) ; the extent of change necessary for POSIX to fulfill the requirement (modification or insertion); and the importance of ultimately standardizing interfaces which meet the requirement (essential, highly desirable, may be deferred, should be reevaluated). Furthermore, for those unfulfilled requirements classified essential or highly desirable, alternatives for achieving standardization (if more than one), and OSSWG recommendations are presented. This section combines all delta information related to each requirement in one centralized place.

Because of the rapidly evolving nature of POSIX, especially the continuous reorganization of unapproved drafts, section 3 does not attempt to cite references to specific chapters, paragraphs, pages, or lines in POSIX documents. Instead, POSIX interfaces are described here using the names commonly used to refer to such interfaces and associated POSIX document (PAR) numbers. Because this document serves not only as an OSSWG working document, but as a reference document for potential NGCR Operating System users, Appendix A lists for each OSSWG requirement, in tabular form, detailed paragraph references to the versions of POSIX documents baselined in section 2, as well as selected tabular information from section 3.

Each unfulfilled OSSWG requirement is coded, both in section 3 and Appendix A, with a rating indicating its significance to the overall NGCR OS interface standardization effort: A rating of "a" indicates that standardization of interfaces which meet the requirement is essential; a rating of "b" indicates that standardization of interfaces which meet the requirement is highly desirable; a rating of "c" indicates that fulfilling this interface requirement can be deferred to a later date; a rating of "d" indicates that the OSSWG should re-evaluate the need for standardized interfaces fulfilling this requirement. All requirements with a rating of "a" or "b" are termed "*significant unfulfilled requirements*", a status which triggers an OSSWG recommendation for fulfilling the requirement as soon as possible.

Section 4, the Big 6 Discussion, offers an overview of the POSIX/OSSWG delta with respect to six major technology areas considered important to the NGCR program in general. This provides an insightful alternative viewpoint on the nature of the delta and how POSIX can be expected to support these technology areas.

In conclusion, Section 5 summarizes the findings of this document.

2. REFERENCES

The following references were used in the preparation of this document:

Ada LRM; "Reference Manual for the Ada Programming Language," ANSI/MIL-STD-1815-A, January 1983.

ISO 9899; "Information Processing System - Programming Languages - C," Published, 1990.

DOD 5200.28-STD; "Department of Defense Trusted Computer System Evaluation Criteria (TCSEC)," December 1985

SECNAV Instruction 5239.2; "Department of the Navy Automated Information Systems (AIS) Security Program," 15 November 1989

OSSWG OCD; "Operational Concept Document for the Next-Generation Computer Resources (NGCR) Operating System Interface Standard Baseline," NUWC Technical Document 10168, February 1993.

IEEE 1003.1; "Information Technology - Portable Operating System Interfaces (POSIX) - Part 1: System Application Program Interface (API) [C language]," Published, December 1990.

IEEE P1003.1a; "Draft Revision to Information Technology - POSIX Part 1 System Application Program Interface (API) [C Language]," Draft 11, February 1994.

IEEE P1003.1e; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) - Amendment #: Protection, Audit, and Control Interfaces [C Language]," Draft 14, March 1994.

IEEE 1003.2; "IEEE Standard for Information Technology - Portable Operating System Interfaces (POSIX) - Part 2: Shell and Utilities," Volume 1 and 2, Published June 1993.

IEEE 1003.2a; "IEEE Standard for Information Technology - Portable Operating System Interfaces (POSIX) - Part 2: Shell and Utilities, User Portability Extension (UPE)," Bundled with IEEE 1003.2 Volume 1 and 2, Published June 1993.

IEEE P1003.2c; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities - Amendment #: Protection and Control Interfaces," Draft 14, March 1994.

IEEE 1003.4; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) - Amendment 1: Realtime Extension [C Language]," Draft 14.1, May 1993.

IEEE P1003.4a; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) - Amendment 2: Threads Extension," Draft 8, October 1993.

IEEE P1003.4b; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) - Amendment x: Realtime System API Extension," IEEE P1003.4b, Draft 8, September 1993.

IEEE 1003.5; "IEEE Standard for Information Technology - POSIX Ada Language Interfaces - Part 1: Binding for System Application Program Interface (API), Published, December 1992.

IEEE P1003.7; "IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - System Administration Interface," Draft 8, February 1992.

IEEE P1003.7.1; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - System Administration Interface/Printing," Draft 6, October 1992.

IEEE P1003.7.2; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 7.2: Software Administration," Draft 10, September 1993.

IEEE P1003.7.3; "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 7.3: Software Management: User and Group Account Management," Draft 2, October 1993.

IEEE P1003.8; "Draft Transparent File Access Revision to Portable Operating System Interface for Computer Environment," IEEE P1003.8, Draft 6, January 1992.

IEEE P1003.12; "Information Technology - Portable Operating System Interface (POSIX) - Part xx: Protocol Independent Interfaces (PII)," Draft 4.1, February 1994.

IEEE P1003.20; "Draft Standard for Information Technology - POSIX Ada Language Interfaces - Part 2: Binding for Realtime Extensions," Draft 2, April 1993

IEEE 1224; "IEEE Standard for Information Technology - Open Systems Interconnection (OSI) Abstract Data Manipulation - Application Programming Interfaces (API) [Language Independent]."

IEEE 1224.1; "IEEE Standard for Information Technology - X.400 Based Electronic Messaging Application Programming Interfaces (API) [Language Independent]."

IEEE 1224.2; "IEEE Standard for Information Technology - Directory Services Application Programming Interface (API) [Language Independent]."

IEEE P1238.1; "Draft File Transfer, Access and Management [FTAM] Application Interface," Draft 1, May 1992

IEEE P1351; "Draft Standard for Information Technology - OSI Application Program Interfaces - ACSE and Presentation Layer Services - Application Programming Interface [Language Independent]," Draft 3, December 1993

IEEE P1353; " Draft Standard for Information Technology - OSI Application Program Interfaces - ACSE and Presentation Layer Services - C Language Binding," Draft 3, December 1993

The following references are relevant to the Delta document but do not at this time directly contribute to its contents:

IEEE P1003.0; "Draft Guide for Information Technology - Portable Operating System Interface (POSIX) - The Open Systems Environment," Draft 16, August 1993.

IEEE 1003.3; "IEEE Standard for Information Technology - Test Methods for Measuring Conformance to POSIX," Published April 1991.

IEEE P1003.13; "Draft Standard for Information Technology - Standardized Application Environment Profile, POSIX Realtime Application Support (AEP)," IEEE P1003.13, Draft 5, February 1992.

3. DETAILED ANALYSIS OF POSIX DELTAS BY REQUIREMENT

This section presents each operating system interface requirement from the OSSWG Operational Concept Document (OCD), grouped according to the same service classes and in the same order as defined in the OCD. For a requirement which is completely fulfilled by POSIX, this section indicates which POSIX interfaces fulfill the requirement, and provides an explanation of how this is accomplished where it isn't completely obvious. For a requirement which is either partially or totally unfulfilled by POSIX, this section describes: the extent of the delta (partial or no POSIX coverage); the extent of change necessary for POSIX to fulfill the requirement (modification or insertion); and the importance of ultimately standardizing interfaces which meet the requirement (essential, highly desirable, may be deferred, should be reevaluated). Furthermore, for those unfulfilled requirements classified essential or highly desirable (the so-called "significant unfulfilled requirements"), alternatives for achieving standardization (if more than one), and OSSWG recommendations are presented.

This section contains frequent references to interfaces and capabilities from the POSIX 1003.1 and 1003.4 standards, as well as the POSIX P1003.4a draft standard. Each of these documents provides a C language binding to the referenced interfaces and capabilities. OSSWG understands that the POSIX 1003.5 standard, the POSIX P1003.20 draft standard, and the Ada LRM provide an Ada language binding to exactly the same set of interfaces and capabilities; however, due to the nature of the bindings and the Ada language itself, identical interfaces and capabilities do not typically have the same nomenclature in the Ada language bindings as in the C language bindings. A further complication is that P1003.20 is currently undergoing a change from "thin" to "thick" binding format. Therefore, this version of the Delta Document will not attempt, in this section, to consistently mention 1003.5 or P1003.20 interfaces whenever 1003.1, 1003.4, or 1003.4a interfaces are cited as fulfilling or partially fulfilling a requirement; this will be undertaken in the next version once P1003.20 has stabilized in its "thick" binding format. Appendix A lists the applicable interfaces and capabilities in both the C language binding documents and the Ada language binding documents.

There is a table presented at the end of each service class with columns marked "Requirement", "Covered", "POSIX Delta", and "Unfulfilled Requirements Rating." The first column contains the OSSWG requirement number. The second column assesses coverage as "Yes", "No", or "Partially." The third column indicates the extent of the POSIX Delta and contains one of the following assessments: "None", "Modification", or "Insertion." "Modification" means that a modification to existing POSIX interfaces would fulfill the OSSWG requirement; "Insertion" means that a modification is not sufficient and that a larger change such as insertion of new interfaces would probably be needed to fulfill the OSSWG requirement. All OSSWG requirements marked as partially or not covered are referred to as "unfulfilled requirements." The fourth column can contain a dash or one of the letters a, b, c, or d. A rating of "a" indicates that standardization of interfaces which meet the requirement is essential; a rating of "b" indicates that standardization of interfaces which meet the requirement is highly desirable; a rating of "c" indicates that fulfilling this interface requirement can be deferred to a later date; a rating of "d" indicates that the OSSWG should reevaluate the need for standardized interfaces fulfilling this requirement. All OSSWG requirements with a rating of "a" or "b" are referred to as "significant unfulfilled requirements", a status which triggers an OSSWG recommendation for fulfilling the requirement as soon as possible.

3.1 GENERAL REQUIREMENTS

These requirements are considered too high level to be covered in this document.

3.2 ARCHITECTURE DEPENDENT INTERFACES

There are no unfulfilled requirements for service class 2. In general, POSIX 1003.1 and 1003.4 support service class 2.

3.2.1 Non-NGCR System Interfaces

Non-NGCR System Interfaces are met by:

1003.1 Process Primitives
1003.1 Input and Output Primitives
1003.4 Process Primitives
1003.4 Input and Output Primitives
1003.4 Shared Memory
1003.4 Message Passing
P1003.12 Network Interface

The OSIF shall support non-NGCR based systems by providing a subset of its services to those systems. The subset of service requests from non-NGCR based systems includes download, initialize, start, resource sharing, process to process message communication, and ability to pass operational status information.

The non-NGCR system may issue service requests over non-NGCR or NGCR network interfaces. The NGCR network interfaces include FUTUREBUS+, SAFENET, (see the operational concept document (OCD), Paragraph 20.8.1.1). The non-NGCR network interfaces include (but are not limited to) VME, MULTIBUS, TCP/IP, RS232, RS422 and 1553B (see OCD paragraph 20.8.2.3).

POSIX does not provide explicit interfacing to non-NGCR networks. However, POSIX can support interfacing to non-NGCR networks given that the term "support" allows for hardware to be added to the non-NGCR network interface, and software to be added to both NGCR and non-NGCR systems. The application implementation of the additional hardware and software will allow the ability to service non-NGCR system service requests.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
2.1	Yes	None	-

3.3 CAPABILITY AND SECURITY INTERFACES

Computer security requirements permeate the engineering process and development environment of a system. The level of security depends on the criticality of the system application and total environment (e.g., physical, procedural, operational, communication, and computer controls). With this in mind, the challenge for the OSSWG and POSIX security working groups has been to create an interface standard that does not preclude meeting the trusted computer systems evaluation criteria (TCSEC) (DoD-STD-5200.28) B3 or A1 class requirements. The approach used to develop the POSIX security standards (P1003.1e and P1003.2c) is similar to the OSSWG security approach where the focus is only on the application program interfaces and commands of the operating system with respect to security, not implementation or assurance details. However, in addressing some of the non interface security concepts, the POSIX subcommittee has tailored these concepts into a POSIX philosophy for uniformity and portability, and documented them in the appropriate P1003.1e and P1003.2c appendixes. The POSIX subcommittee has been very effective, thus far, in addressing the nonsupported, security-related concepts without mandating a specific design or architecture. Those areas that are not supported by P1003.1e are discussed in its appendix B, the unsupported security section. This allows a contractor design and development flexibility, while still providing the basic conformity and interface consistency found in standards. The POSIX Distributed Security Study Group (1003.22), was convened in early 1992 to examine security standardization issues which fall outside the domain of P1003.1e and P1003.2c.

They will be assessing existing work in this area and analyzing the potential for standardization of distributed security work, and will draft a Guide similar to P1003.0 to this effect.

As stated, some requirements are not interface-specific security issues but, rather, system-unique security issues. In this case, the interpretation of the TCSEC requirements in conjunction with the system security policy would take precedence over any aspects of the P1003.1e and P1003.2c interface standards. Unfulfilled requirements will need to be analyzed by the system engineering office within the context of the system being developed. Each requirement should be studied to determine its applicability to the system and, if required, the suitability of the contractor's design in context of the system's security policy. Therefore, the capability to fulfill these requirements should be deferred until a need is determined and how they may be best implemented to satisfy the system's security policy.

Of the 24 OSSWG requirements, 21 are addressed by the P1003.1e and P1003.2c standards (see Requirements Coverage Summary). [Note: The P1003.1e standard addresses all 24 OSSWG security requirements in different ways. Some of the requirements are in the interface section, while others are addressed in appendix B as non interface, nonsupported security mechanisms. The OSSWG agrees with the P1003.6 standards committee on this format; thus, OSSWG feels the committee has sufficiently addressed the OSSWG requirements.]

In assessing the OSSWG security requirements, it was determined that the following requirements are not addressed by the P1003.1e or P1003.2c standards: Object Reuse (3.17), Trusted Path (3.23), and Trusted Recovery (3.24). However, some of these requirements are within the scope of P1003.22, and while they constitute implementation concerns, they could receive attent'. in the P1003.22 Guide to the POSIX Security Framework. However, the 1003.22 working grou, will not produce a standard, per se, but rather a guide to distributed security issues. Therefore, it is recommended that these requirements be dropped from consideration of any API standard approved by the NGCR OSSWG.

3.3.1 Audit Data Storage

The capability and security interfaces service class requirements are addressed in the P1003.1e document. This OSSWG requirement is covered in the interface portion of the standard.

3.3.2 Audit Generation

Refer to 3.3.1.

3.3.3 Audit Record Contents

Refer to 3.3.1.

3.3.4 Audit Data Manipulation

Refer to 3.3.1.

3.3.5 Device Labels

Refer to 3.3.1.

3.3.6 Basic DAC

Refer to 3.3.1.

3.3.7 DAC Inclusion/Exclusion

The requirement for DAC Inclusion/Exclusion (3.7) is met by studying the functionality of the interface, but the document does not provide a clear discussion of exclusion.

3.3.8 DAC Propagation

Refer to 3.3.1.

3.3.9 Labeling of Export Channels

Refer to 3.3.1.

3.3.10 Setting Communication Labels

Refer to 3.3.1.

3.3.11 Identification and Authentication

This OSSWG requirement is addressed by the *P1003.1e standard in appendix* B. Even though it specifies this requirement as an unsupported security mechanism, the standard does not preclude satisfying this requirement; this requirement is addressed in DoD Standard 5200.28.

Note that 1003.1 also provides interfaces to identify and to inquire about the identity of users.

3.3.12 Labeling of Human Readable Output

Refer to 3.3.1.

3.3.13 Subject and Object Labeling

For Subject and Object Labeling (3.13), the POSIX definition of subjects and objects is very broad and may not provide sufficient detail to meet B2 requirements and above. However, for the purpose of an interface standard this should be acceptable because significant depth in this area will be provided by either the vendor or the contractor as the system architecture and design that incorporate the interface standard are developed.

3.3.14 Label Contents

Refer to 3.3.1.

3.3.15 MAC Policy

Refer to 3.3.1.

3.3.16 MAC Manipulation

MAC Manipulation has been addressed in P1003.1e and P1003.2c, while the manipulation of labels remains a non-programmatic, non-interface issue dictated by the security policy.

3.3.17 Object Reuse

This unfulfilled requirement is classified as "d" (re-evaluate).

Requirement: The OSIF shall provide that all objects are sanitized prior to allocation to a user.

Description of the Delta: Object Reuse is not a programmatic interface-related requirement. It is a requirement between a user terminal, peripheral hardware elements, and the operating system's trusted computing base. A conforming implementation may implement a strong object reuse policy without impacting the API specified by the standard.

Recommendation: The requirement will be levied on the developers through the TCSEC when required; thus, no further action is recommended.

3.3.18 User Notification of Sensitivity Label

Refer to 3.3.11.

3.3.19 Sensitivity Label Query

Refer to 3.3.11.

3.3.20 System Integrity

Refer to 3.3.11.

3.3.21 Identification of Users Based on Roles

Refer to 3.3.11.

3.3.22 Least Privilege

Refer to 3.3.1.

3.3.23 Trusted Path

This unfulfilled requirement is classified as "d" (re-evaluate).

Requirement: The OSIF shall support a trusted communication path between the user and the system, activated exclusively by the user.

Description of the Delta: The trusted path requirement is not a programmatic interface-related requirement. It is a requirement between a user terminal and the operating system's trusted computing base. It will be addressed in the P1003.22 Guide to the POSIX Security Framework, which will complement the work being done in both the 1003.0 and 1003.6 committees.

Recommendation: The requirement will be levied on the developers through the TCSEC when required; thus, no further action is recommended.

3.3.24 Trusted Recovery

This unfulfilled requirement is classified as "d" (re-evaluate).

Requirement: The OSIF shall provide procedures or mechanisms or both to assure that, after a discontinuity, recovery without a protection compromise is obtained.

Description of the Delta: This is not an programmatic interface related requirement but a requirement internal to the trusted computing base (TCB) concerned with trusted recovery to a secure state of the TCB when non recoverable failure occurs. It will be addressed in the P1003.22 Guide to the POSIX Security Framework, which will complement the work being done in both the 1003.0 and 1003.6 committees.

Recommendation: The requirement will be levied on the developers through the TCSEC when required; thus, no further action is recommended.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
3.1	Yes	None	-
3.2	Yes	None	-
3.3	Yes	None	-
3.4	Yes	None	-
3.5	Yes	None	-
3.6	Yes	None	-
3.7	Yes	None	-
3.8	Yes	None	-
3.9	Yes	None	-
3.10	Yes	None	-
3.11	Yes	None	-
3.12	Yes	None	-
3.13	Yes	None	-
3.14	Yes	None	-
3.15	Yes	None	-
3.16	Yes	None	-
3.17	No	Insertion	d
3.18	Yes	None	-
3.19	Yes	None	-
3.20	Yes	None	-
3.21	Yes	None	-
3.22	Yes	None	-
3.23	No	Insertion	d
3.24	No	Insertion	d

3.4 DATA INTERCHANGE INTERFACES

Appendix B, Rationale and Notes, of the 1003.1 indicates that the POSIX groups felt the issue of data format should be addressed in 1003.1. 1003.1/1003.5 does not yet provide a standard data interchange interface, nor does it define a standard format for the data. 1224 is developing an ASN.1 (Abstract Syntax Notation One) API. A notable hole in the 1224 work is a result of the working group decision not to provide interfaces for floating-point data.

A non-POSIX alternative for meeting the data interchange requirement is XDR (External Data Representation), an Internet standard (see RFC1014). XDR is well-established, provides a relatively straight-forward binding to P1003.12, is capable of supporting realtime communication, is canonical, has no explicit typing, and represents arbitrary data structures in a consistent, well-documented manner. However, XDR at this time does not have POSIX or ISO support.

Data Interchange Interfaces are necessary to support the Big 6 requirement for heterogeneity.

One aspect of the Data interchange issue arises from the fact that the various hardware and software platforms used in Navy systems represent various uncoordinated data types being passed between many systems. These systems were developed on essentially the same computer hardware, at different times, by different vendors, and for different sponsors, with incremental funding. Most of these systems were developed long ago, prior to any formal standardization process, and were designed to perform specific tasks that were not always integrated. The cost of ownership of this wide spectrum of systems is inconsequential compared with the replacement cost of upgrading to systems that have a standardized data interchange. Therefore, an interface is needed to support the required "normalized" representations of data interchanged between these different systems. This interface would provide standards for upgrading these older systems with a more effective approach.

Likewise, the interface would provide standards for combining COTS products effectively, whether or not the products originate in older systems.

3.4.1 Data Interchange Services (Data Format Conversion)

This unfulfilled requirement is classified as "a" (essential).

Requirement: The OSIF shall support an access to services that perform data conversion.

Description of Delta: P1351 and P1353 have developed an ASN.1 API. However, this API will not support floating-point data. ASN.1 is already an ISO standard. It is canonical, supports explicit typing, and represents arbitrary data structures in a consistent, well-documented manner. A potential disadvantage of ASN.1 is that it may not be capable of supporting realtime systems.

Resolution Alternatives:

1. Pursue adding floating-point data support to P1351 and P1353.
2. Pursue standardizing XDR within POSIX.
3. Adopt XDR as another OSSWG-recommended standard (in addition to POSIX).

Recommendation: Investigate 1. and 2.; should 2. fail, pursue 3. to meet realtime requirements.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
4.1	Partially	Modification	a

3.5 EVENT AND ERROR INTERFACES

In general, the POSIX standards support service class 5 (Event and Error Interfaces) in a rudimentary way. There are three areas that are not complete:

1. Basically, POSIX provides reactive error management while OSSWG requires proactive behavior. Attempting to support proactive requirements on top of a reactive interface will result in performance penalties. The existing (proactive) services are highly-oriented toward providing event services (via the "signal" concept) while downplaying error reporting.

2. POSIX currently does not have a consistent error handling strategy. The POSIX working groups covering distribution are beginning to develop such a strategy.

3. POSIX does not provide adequate coordination and recording services.

While none of the requirements in service class 5 are completely satisfied by POSIX interfaces, all the associated OSSWG requirements remain necessary for Navy systems. Given that the OSSWG will now deal only with APIs for the OSIF, requirement 5.1 becomes deferred for errors, since the error information comes from sources other than applications; it is fulfilled in the case of events other than errors.

POSIX signals provide a useful abstraction for managing asynchronous events and can be used to coordinate the activities of processes. In particular, signals unify the following:

- synchronous exceptions, such as floating point overflow, division by zero, and invalid addresses or instructions
- abortion of a process or thread of control
- suspension of a process
- time-outs such as an alarm or timer expiration
- asynchronous notification from one process or another of an application-specific event that demands attention

However, precisely because they are so all-encompassing, signals also:

- confuse synchronous traps with asynchronous events
- can be aliased in confusing ways
- can be lost
- are unique resources which cause problems when various independent application components are integrated

Conflicts over the right to handle a signal are a problem for the Ada runtime, since it requires the use of certain specific signals, and it is not something a user can ordinarily be expected to patch up. The POSIX Ada bindings address this situation by denying an Ada application the ability to handle certain signals which are expected to be used by the Ada runtime system. This still leaves the need for intervention if an Ada application wants to use a C language library that depends on catching the same signals used by the Ada runtime system.

The OSSWG requirements could be met by adding new interfaces to POSIX. Existing interfaces do not need to be modified or deleted. However, some philosophical views and assumptions of the POSIX community differ considerably from the OSSWG conceptual model.

Examples are access to hardware interrupt masks and error logging. Both were cited as "out of scope" by the POSIX community.

1003.4b has developed interrupt control interfaces which fulfill Requirement 5.5 and contribute to the fulfillment of Requirement 5.2. Due to hardware dependencies, it may not be appropriate to attempt to standardize interfaces for masking/unmasking interrupts.

__Executive Summary:__ The following paragraphs serve as an explanation and summary of section 3.5, Event and Error Interfaces, and section 3.11, Reliability, Adaptability, and Maintainability Interfaces. While these two service classes are closely related, note that service class 5 goes beyond strictly error interfaces, which also apply to service class 11, and deals more broadly with events, which may or may not be related to errors. The thrust of this summary is system fault and error management, which is concerned with the error aspects of service class 5 and with service class 11. Section 3.5 does, however, also discuss events in detail.

In addition, while some of the requirements from service classes 5 and 11 deal with interfaces between an operating system and entities other than application software, this summary and sections 3.5 and 3.11 consider satisfying requirements only through an API. The discussion of other types of operating system interfaces is deferred at this time.

In general, the OSSWG discovered that POSIX provides or supports little in the way of interfaces for service classes 5 and 11 as they relate to system fault and error management. (Sections 3.5 and 3.11 discuss the deltas between what the OSIF requires and what POSIX supplies for each OSIF requirement in detail.) Consequently, the OSSWG considered the following alternatives to resolve the deltas between the OSIF requirements and POSIX:

1. Enhance existing POSIX interfaces to include this capability. System fault and error management is not generally a natural extension to existing POSIX interfaces. It may fit as new work under POSIX 1003.7, system administration.

2. Submit a new POSIX PAR to do this work. POSIX may require a new PAR even should this work be done under 1003.7. A substantial body of existing practice is available for system fault and error management in current military tactical systems and may also be available in such commercial applications as telephone, medical, and banking systems. The availability of people to do this work may well be the deciding factor in providing this capability in POSIX. People would probably have to come largely from OSSWG as general interest in the POSIX community for this kind of activity seems to be low. However, the OSSWG should also contact commercial parties where interest may be growing.

3. Mature a standard outside of POSIX. UNIX International (High Availability Investigative Team), Open Software Foundation (OSF), and X3T8 (Fault Isolation) have efforts that might fill a large number of the current deltas. OSSWG could use these as the vehicles to mature a industry standard outside of POSIX. At the appropriate time a new PAR could be pursued in POSIX.

4. Develop a new military standard. This is a less acceptable alternative than 2, although comparable in effort, because it is external to the OSIF baseline.

5. Levy the requirements and the OSIF general requirements (e.g., modularity, extensibility, uniformity) on vendors but do not provide a standard as such. This alternative relies on vendors to develop some commercial existing practice in this area on which to potentially standardize at a later date.

The OSSWG recommends at this time that a standard be matured outside of POSIX, through UNIX International, OSF, and X3T8 as appropriate. Unfulfilled OSIF requirements which could be satisfied by other efforts include:

3.5.1 Event and Error Receipt
3.5.2 Event and error distribution
3.5.3 Event and error management
3.5.4 Event logging
3.11.1 Fault information collection
3.11.2 Fault information request
3.11.3 Diagnostic tests request
3.11.4 Diagnostic tests results
3.11.5 Operational status
3.11.6 Fault detection thresholds
3.11.7 Fault isolation
3.11.8 Fault response
3.11.9 Reconfiguration
3.11.10 Enable/disable system component
3.11.11 Performance monitoring
3.11.12 Set resource utilization limits
3.11.13 Resource utilization limits violation

The OSSWG recommends satisfying 3.5.6, Mask/Unmask Interrupts in P1003.4b where this work has already been undertaken. Mask/Unmask Interrupts is not provided by P1003.4b because of hardware dependencies (classification as a significant unfulfilled requirement should be reconsidered.) Additionally, some minimal functionality can be achieved for requirements 3.11.3, Diagnostic Test Requests, 3.11.4, Diagnostic Test Results, 3.11.5, Operational Status, 3.11.8, Fault Response, 3.11.9, Reconfiguration, and 3.11.10, Enable/Disable System Component through interface service devctl() in P1003.4b. Devctl() allows standard access to 'non standardized' hardware devices.

3.5.1 Event and Error Receipt

This *unfulfilled requirement* is classified as "c" (may be deferred).

OSSWG requirement 5.1 is partially covered by POSIX. While the event interfaces exist, and error interfaces are provided for individual processes, there are no error coordination or distribution interfaces.

<u>Requirement:</u> The OSIF shall support the receipt and coordination of event and error information.

<u>Description of Delta:</u> This requirement refers to error information coming into the OS across the OSIF other than through an API for subsequent distribution according to requirement 3.5.2. The event receipt part of this requirement is met by the POSIX Signals interface and the Interrupt Control interfaces in P1003.4b.

<u>Recommendation:</u> For error receipt, because OSSWG is only concerned with the API portion of the OSIF at this time and for most applications this requirement deals with parts of the OSIF other than APIs, this requirement delta is a low priority. Monitor and participate in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate portions of standards into POSIX.

3.5.2 Event and Error Distribution

This *unfulfilled requirement* is classified as "a" (essential).

P1003.4b Interrupt Control specifies that, upon occurrence of a designated interrupt, a designated process or thread is to be notified, or a designated user-written Interrupt Service Routine (ISR) is to be executed (or both). This interrupt control capability, in conjunction with 1003.1/1003.4/1003.4a signals, would provide some coverage of requirement 5.2 (distribution of event and error information). In particular, the interrupt control mechanism could be used for the distribution of information on events and errors resulting in hardware interrupts (such as hardware device errors). However, this distribution mechanism would not be applicable to certain operating system errors (such as those in which kernel data structures become faulty).

Another possible deficiency in the coverage of requirement 5.2 is the fact that functions return indication of only a single error, instead of all errors that occur during function processing.

Requirement: The OSIF shall provide for the selective distribution of event and error information.

Description of Delta: POSIX 1003.1 provides for the distribution of events through signals. Table 3-1 (1003.1) lists the signals that all POSIX implementations must support and Table 3-2 (1003.1) lists those signals that a system implementing job control must support. However, "an implementation may define additional signals that may occur in the system" (1003.1). For particular systems, it may be significant that the signals defined by 1003.1 and 1003.5 do not allow for any user-defined information, such as a pointer to an error report, to be passed with the signal and do not queue multiple occurrences of a signal. The Signals interface is enhanced in 1003.4 with the addition of Queued Signals, and all signal types are extended to threads in P1003.4a. The 1003.4 and P1003.20 specifications allow an application to reserve a range of signal numbers as real-time signals. These signals may pass a user-defined value or pointer to the signal-catching function. In addition multiple occurrences of real-time signals are queued for the application in FIFO order.

POSIX provides for the distribution of errors to the requesters of individual functions. Each function specifies which errors all POSIX implementations must detect and which are optional. 1003.1 lists the possible errors. However, "implementations may support additional errors not included in this clause, may generate errors included in this clause under circumstances other than those described in this clause, or may contain extensions or limitations that prevent some errors from occurring" (paragraph 2.4, 1003.1). "If more than one error occurs in processing a function call, this part of ISO/IEC 9945 does not define in what order the errors are detected; therefore, any one of the possible errors may be returned" (paragraph 2.4, 1003.1). [CAN THIS APPROACH BE TOLERATED?] In addition, realtime extensions in POSIX 1003.4b provide for handling of interrupts. In 1003.4b the occurrence of an interrupt can be made to notify a process or thread, or start the execution of a user-written ISR (or both).

The OSIF requires that all possible errors be available, not just one of those possible. [AGAIN, CAN THIS BE TOLERATED?] It also requires that there be a means for coordinating the distribution of errors, as for example to a single process responsible for error analysis. The 1003.4b interrupt control interface enables distribution of certain errors, namely those resulting in hardware interrupts. Besides the fact that the P1003.4b Interrupt Control interface can deliver only hardware interrupts and the Signals interface can deliver any event or error defined by the system, it may be important for particular systems to note another difference between the two interfaces: Interrupt Control has distinct registration/deregistration functions for each interrupt whereas the Signals interface relies on signals be sent to or retrieved by the proper application software.

Recommendation: The OSSWG recommends continued support for approval of the P1003.4b Interrupt Control interfaces via the balloting process.

And, to completely satisfy this requirement, OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. The POSIX Services for Reliable, Available, and Serviceable Systems group, in addition to proposing new interfaces, may suggest modifications to existing interfaces, such as reserving a set of real-time signal numbers for error reporting. When these groups develop mature standards, move appropriate standards into POSIX.

3.5.3 Event and Error Management

This unfulfilled requirement is classified as "a" (essential).

Requirement: The OSIF shall support the timely delivery of interrupt and asynchronous events to system components and shall support the implementation of user-selectable error processing alternatives. Alternatives shall include, as a minimum, filtering, retry, ignore, and accumulate occurrences.

Description of Delta: POSIX does make special provisions for the timely delivery of interrupts and asynchronous events which generate interrupts to system components; P1003.4b Interrupt Control interfaces provide for process or thread notification on occurrence of an interrupt and/or for handling the interrupt via an Interrupt Service Routine (ISR). For asynchronous events which generate signals, "Implementations should deliver unblocked signals as soon after they are generated as possible. However, it is difficult for POSIX.1 to make specific requirements about this, beyond those in kill() and sigprocmask(). Even on systems with prompt delivery, scheduling of higher priority processes is always likely to cause delays" (paragraph B.3.3.1.2, 1003.1).

Within the limits discussed under requirement 3.5.2 (i.e., POSIX does not provide for coordination in the distribution of events and errors), some user-selectable error processing alternatives are available. Processes can mask signals (paragraph 3.3.1.2, 1003.1). Processes can also choose among three types of actions that they can associate with a signal: a default action, ignore, and a signal catching function (paragraph 3.3.1.3, 1003.1). Retries and accumulation of occurrences would then be the responsibility of the individual processes. In particular, occurrences of a particular event or error could not be collected or action taken on behalf of several processes or on behalf of the system as a whole through the interface.

Recommendation: OSSWG recommends continued support for approval of the P1003.4b Interrupt Control interfaces via the balloting process. If necessary, OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; the POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate standards into POSIX.

3.5.4 Event Logging

This unfulfilled requirement is classified as "a" (essential).

Requirement 5.4, event logging, is not currently supported by POSIX.

Requirement: The OSIF shall support logging events to application-defined storage. The types of events and event sources shall be dynamically selectable/deselectable.

Description of Delta: POSIX does not support logging events. The 1003.4 working group considers this to be a system administration issue.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; the POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate standards into POSIX.

3.5.5 Enable/Disable Interrupts

This requirement is directly met by the Interrupt Control interfaces in P1003.4b. These interfaces provide for mutual exclusion between application code and Interrupt Service Routine (ISR) code, effectively providing the functionality of Enable/Disable Interrupts in a generalized interface which permits implementations for both uni-processor and multi-processor systems.

3.5.6 Mask/Unmask Interrupts

This unfulfilled requirement is classified as "c" (may be deferred).

Requirement: The OSIF shall provide the ability to mask and unmask interrupts. Note that this requirement has particular relevance for Ada applications, as specified in paragraph 3.16.18. Changes to the recommendations should take that fact into account.

Description of Delta: Within the limits discussed under requirement 3.5.2 (i.e., POSIX does not provide for the collection and coordination of all events and errors), POSIX provides the ability to mask and unmask events through its signal processing (1003.1). Therefore, complete resolution of the deltas for this requirement depend on the resolution of requirement 3.5.2.

While POSIX does currently provide the capability to handle interrupts in P1003.4b, the interfaces therein do not provide the capability to mask and unmask interrupts. Hardware dependencies make it inappropriate to standardize such interfaces.

Recommendation: We recommend that the OSSWG view the masking and unmasking of interrupts as inappropriate for standardization.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
5.1	Partially	Insertion	-/c
5.2	Partially	Insertion	a
5.3	Partially	Insertion	a
5.4	No	Insertion	a
5.5	Yes	None	-
5.6	Partially	Insertion	c

3.6 FILE INTERFACES

In general, the POSIX standards support service class 6 in a substantially complete way. The information that follows was primarily derived from 1003.1, 1003.4, P1003.4a, and P1003.4b documentation.

If you use Ada Direct_IO over POSIX files, then the 1003.5 Change_Working_Directory operation in package POSIX_Process_Environment should be done at system initialization to establish the default working directory.

The requirements for: Contiguous Read of a File (6.1), Protect an Area Within a File (6.2), File Management Suspend/Resume for Process (6.4), File Management Block Requests (6.5), Create (6.16), Open (6.7), Point within a file (6.8), Read (6.9), Write (6.19), Write Contiguous (6.20), Close (6.10), Delete a file (6.11), Create (6.12), Specify Default (6.13), Delete directories (6.14), and Query or Modify File attributes (6.17 - 6.18) are directly met by a combination of 1003.1, 1003.4, and P1003.4b. Shadow Files (6.15) is met by the interfaces listed above in combination with resource locking and/or mutual exclusion interfaces provided by 1003.4 and P1003.4a.

The requirement for: File Management Scheduling (6.3) is not met or insufficiently met by POSIX. File Management Scheduling requires a method to specify a response time for file requests. POSIX does not include this as part of the file interface.

Note that both Ada and POSIX define file operations. The two I/O systems are not based on identical file models. The POSIX I/O system has the objective of making the POSIX I/O model available to the user. With both sets of I/O operations available, it is possible that a given collection of application programs will use both sets of operations. For this reason, it is desirable to permit the interchange of external files so that they can be read and updated by the use of either set of I/O operations after being created and written by a different set of I/O operations. Thus, POSIX extends the Ada file model in several useful ways, including:

- a hierarchical, persistent file name-space
- file/device control
- memory mapping (of files)
- standard error-output file
- appending to a sequential file
- files with records of mixed types and sizes

The POSIX I/O system does not have the objective of incorporating all the functionality of the Ada I/O model. Instead, it interprets relevant portions of the Ada LRM and constrains and details some of the implementation dependencies permitted by the Ada LRM so that Ada I/O is more completely defined in a POSIX environment. Thus, the POSIX I/O model fits the Ada I/O model fairly well.

Unfortunately, a complete mapping between the POSIX and Ada I/O operations is quite difficult, primarily because of the lack of underlying standardization concerning external representations of data. On a POSIX system, Ada external files are implemented as POSIX files, but the view of a file via the Ada I/O packages is different from the view via the POSIX interfaces. There is also a difference between portable character sets, though this is likely to be reduced in Ada 9X. Furthermore, the combination of POSIX and Ada files does create the possibility of some new errors. In general, the effects of interleaved Ada and POSIX operations on the same open file are unpredictable. The POSIX Ada binding provides a way to open an Ada file object with a specified POSIX file descriptor, but states that the effect is implementation-defined.

3.6.1 Contiguous Read of a File

This requirement is directly met by a 1003.1 Input/Output Primitives; 1003.4 Asynchronous or List Directed I/O and Memory Mapped Files; and P1003.4b Advisory Information.

3.6.2 Protect An Area Within A file

This requirement is directly met by 1003.1 open() and File Control; and 1003.4 Memory Mapped Files.

3.6.3 File Management Scheduling

This unfulfilled requirement is classified as "c" (may be deferred).

Requirement: The OSIF shall support a capability to specify a response requirement for the service being requested for file management.

Requirement Rationale: For hard deadline real-time systems, the file manager must schedule service processing based on the response requirements of the requests submitted by the users. FIFO scheduling is unacceptable for real-time applications. The file manager must also support the notion of preemption.

Description of Delta: POSIX does not require a method for specifying a response time for scheduling I/O.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. The 1003.4 working group is unable to identify existing practice for such interfaces, and thus does not consider this appropriate for standardization under the 1003.4 charter.

2. Submit a new POSIX PAR to do this work. The availability of people to do this work is questionable. People would probably have to come largely from OSSWG as general interest in the POSIX community for this kind of activity seems to be low.

3. Assume a standard outside POSIX. No standards that answer this kind of requirement are apparent at this time.

4. Develop a new military standard. This is a less acceptable alternative than 2 because it is external to the OSIF baseline. At the same time, it suffers from the same handicaps as 2, lack of people to do the work

5. Levy the requirements on vendors without a standard imposed. This alternative relies on vendors to develop some commercial existing practice in this area on which to potentially standardize at a later date.

Recommendation: Based on alternative 1, we recommend that the OSSWG view File Management Scheduling as inappropriate for standardization.

3.6.4 File Management Suspend/Resume for Processes

This requirement is directly met by 1003.1 blocking and non-blocking open() and write(); 1003.4 Asynchronous I/O; and P1003.4b Device Control.

3.6.5 File Management Block Requests

This requirement is directly met by 1003.1 read(), write(), and lseek(); 1003.4 Memory Mapped Files; and P1003.4b Advisory Information.

3.6.6 Round Robin File Management

This requirement has been deleted.

3.6.7 Open a File

This requirement is directly met by 1003.1 open(); 1003.4 open(); and P1003.4b Advisory Information.

3.6.8 Point Within a File

This requirement is directly met by 1003.1 lseek(); and 1003.4 Memory Mapped Files.

3.6.9 Read a File

This requirement is directly met by 1003.1 read(); and 1003.4 Asynchronous or List Directed Read and Memory Mapped Files.

The Ada standard Direct_IO package will be provided as part of standard Ada. This package contains two READ file operations. The input parameters for the first read operation include the FILE identifier and the index to read FROM the file. The second read operation is an overloaded version of the first without the parameter identifying the index to read FROM. The only output parameter for both read operations contains the ITEM to be read.

3.6.10 Close a File

This requirement is directly met by 1003.1 close().

The Ada standard Direct_IO package will be provided as part of standard Ada. This package contains a CLOSE file operation. The only parameter is both input and output and is the FILE identifier.

3.6.11 Delete a File

This requirement is directly met by 1003.1 unlink(); and 1003.2 "rm."

The Ada standard Direct_IO package will be provided as part of standard Ada. This package contains a DELETE file operation. The only parameter is both input and output and is the FILE identifier.

3.6.12 Create a Directory

This requirement is directly met by 1003.1 mkdir(); and 1003.2 "mkdir."

3.6.13 Specifying Default Directory

This requirement is directly met by 1003.1 chdir(); and 1003.2 "cd."

3.6.14 Delete a Directory

This requirement is directly met by 1003.1 rmdir(); and 1003.2 "rmdir."

3.6.15 Shadow Files

This requirement is "shall support" and is thus met by the interfaces listed above in combination with resource locking and/or mutual exclusion interfaces provided by 1003.4 and P1003.4a. However, because these interfaces do not necessarily provide sufficient support to maintain shadow files at several nodes of a distributed system, this delta must be carefully re-evaluated if this requirement is modified to explicitly call out distributed shadow file support.

3.6.16 Create a File

This requirement is directly met by 1003.1 open() and creat().

The Ada standard Direct_IO package will be provided as part of standard Ada. This package contains a CREATE file operation. The only input output parameter is the FILE identifier. The input parameters include the MODE, file NAME, and a FORM parameter. The MODE parameter identifies the file as read only, write only, or both read and write. The file NAME is a string identifying the name of the file. The FORM parameter is a string which is user defined. The POSIX_Supplement_To_Ada_IO defined in 1003.5/8.2 will be used to build a POSIX-compliant FORM parameter.

3.6.17 Query File Attributes

This requirement is directly met by 1003.1 stat(), fstat(), access(), and lseek().

3.6.18 Modify File Attributes

This requirement is directly met by 1003.1 chmod(), chown(), utime(), and lseek(); 1003.4 ftruncate(); and P1003.4b Advisory Information. Also, P1003.2 provides the "chmod" shell command to meet this requirement.

3.6.19 Write a File

This requirement is directly met by 1003.1 write(); and 1003.4 Asynchronous or List Directed Write and Memory Mapped Files.

The Ada standard Direct_IO package will be provided as part of standard Ada. This package contains two WRITE file operations. The input parameters for the first write operation include the FILE identifier and the index to write TO the file. The second write operation is an overloaded version of the first without the parameter identifying the index to write TO. The only output parameter for both write operations contains the ITEM to be written.

3.6.20 Write Contiguous File

This requirement is directly met by 1003.1 write(); 1003.4 Asynchronous or List Directed Write and Memory Mapped Files; and P1003.4b Advisory Information.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
6.1	Yes	None	-
6.2	Yes	None	-
6.3	No	Insertion	c
6.4	Yes	None	-
6.5	Yes	None	-
6.6	Deleted	None	Deleted
6.7	Yes	None	-
6.8	Yes	None	-
6.9	Yes	None	-
6.10	Yes	None	-
6.11	Yes	None	-
6.12	Yes	None	-
6.13	Yes	None	-
6.14	Yes	None	-
6.15	Yes	None	-
6.16	Yes	None	-
6.17	Yes	None	-
6.18	Yes	None	-
6.19	Yes	None	-
6.20	Yes	None	-

3.7 GENERALIZED I/O INTERFACES

In general, the POSIX standards support service class 7, generalized I/O, in a substantially complete way. This is assuming the definition of a "file" found in 1003.1 section 2.3, includes any and all devices. This means that any device can be represented by a file.

3.7.1 Device Driver Availability

This unfulfilled requirement is classified as "a" (essential).

This requirement for device driver availability (7.1) is not met by POSIX and is considered by POSIX to be implementation dependent.

Requirement: The OSIF shall provide the interfaces necessary to support the addition of device drivers.

Description of Delta: P1003.4b Interrupt Control allows application servicing of device interrupts. 1003.4 mmap() allows devices to be memory mapped, but only for devices currently known to the system as special files. Not all operating system services typically required by a device driver are shown at the POSIX interface (e.g. mapping a user buffer to a DMA address).

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. This requirement could be inserted into P1003.7 system administration. In the P1003.7 document, place holders exist for interfaces which would be the same type of interfaces needed for device drivers.

2. Assume a standard outside of POSIX. The IEEE P1256 OBIOS group is presently working on standardizing device driver interfaces. The applicability of this standard to the OSSWG operating system standard needs to be investigated.

3. OSSWG defined based on existing practice.

Recommendation: Alternative 1 should be pursued.

3.7.2 Open Device

This requirement is met directly by 1003.1 General File Creation, and 1003.5 Creating and Removing Files.

3.7.3 Close Device

This requirement is met directly by 1003.1 File Descriptor Deassignment, and 1003.5 Close.

3.7.4 Transmit Data

This requirement is met directly by 1003.1 Write, 1003.4 Asynchronous Write, 1003.4 List Directed I/O, and 1003.4 Memory Mapping of special files (devices). The Ada interfaces appropriate for transmitting data include 1003.5 Write and Generic Write. The Ada generic write allows the user to identify a data type appropriate for the data which is sent.

3.7.5 Receive Data

This requirement is met directly by 1003.1 Read, 1003.4 Asynchronous Read, 1003.4 List Directed I/O, and 1003.4 Memory Mapping of special files (devices). The Ada interfaces appropriate for receiving data include Read and Generic Read. The Ada generic read allows the user to identify a data type appropriate for the data which is received.

3.7.6 Device Event Notification

This unfulfilled requirement is classified as "a" (essential).

Device Event Notification is a compound requirement comprising requirements 5.1 (Event and Error Receipt), 5.2 (Event and Error Distribution), 5.3 (Event and Error Management), 5.4 (Event Logging), 5.5 (Enable/Disable Interrupts), and 5.6 (Mask/Unmask Interrupts) applied specifically to events, errors, and interrupts originating at a peripheral device. It remains unfulfilled to the extent that any of its dependent requirements remains unfulfilled for devices. Refer to section 3.5, Event and Error Interfaces for specific information on those requirements.

3.7.7 Control Device

This requirement is directly met by 1003.1 Control Operations on Files, 1003.1 General Terminal Interface, 1003.5 File Control, and 1003.4b Device Control.

3.7.8 I/O Directory Services

The requirement for I/O directory services is met directly by 1003.1 Files and Directories, P1003.1a File Hierarchy Streams, and 1003.5 Packages POSIX_Files and POSIX_File_Status.

3.7.9 Device Management Suspend/Resume for Processes

This requirement is fully met by 1003.1 Open a File, 1003.1 Read from a File (device), 1003.1 Write to a File (device), 1003.4 Asynchronous Input and Output, P1003.4b Device Control, and 1003.5 Read, Write, Generic Read, Generic Write.

3.7.10 Mount/Dismount Device

This unfulfilled requirement is classified as "a" (essential).

This requirement is not met by POSIX and is considered by POSIX to be implementation dependent.

Requirement: The OSIF shall support the capability to mount and dismount a logical or physical device.

Description of Delta: Not presently shown at POSIX Interface

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. This requirement could be inserted into P1003.7 system administration. In the P1003.7 document, place holders exist for interfaces which would be the same type of interfaces needed for mounting and dismounting a device. This was deferred to P1003.7.5 for which no draft has yet been generated.

2. OSSWG defined based on existing practice.

Recommendation: Insert into a P1003.7 system administration document.

3.7.11 Initialize/Purge Device

This requirement is directly met by 1003.1 General Terminal Interface, and P1003.4b Device Control.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
7.1	Partially	Insertion	a
7.2	Yes	None	-
7.3	Yes	None	-
7.4	Yes	None	-
7.5	Yes	None	-
7.6	Partially	Insertion	a
7.7	Yes	None	-
7.8	Yes	None	-
7.9	Yes	None	-
7.10	No	Insertion	a
7.11	Yes	None	-

3.8 NETWORK AND COMMUNICATIONS INTERFACES

In general, the POSIX Standards partially support Service Class 8, Network and Communication Interfaces. Most of the input to the evaluation of this service class is derived from the 1003.12 protocol independent interfaces working group, the 1238 ACSE and Presentation API working group, the 1238.1 FTAM API working group, the 1224.2 directory services API working group, and the Realtime Distributed Systems Communication API (1003.21) project. The Realtime Distributed Systems Communication study group first met in July, 1992, as a 1003.12 splinter group; it submitted a PAR as a separate POSIX working group in fall, 1992. The PAR was subsequently approved as 1003.21.

The 1003.12 working group is developing two levels of networking interfaces. One is the Simple Network Interface (SNI). The other is the Detailed Network Interface (DNI). DNI will have two C bindings, Berkeley sockets and X/Open's XTI (the standardized version of AT&T's Transport Layer Interface (TLI)). The two C bindings position is a compromise resulting from the controversy over whether to choose sockets, XTI, or a third interface made up of elements from both sockets and XTI as the DNI.

1003.21 plans to develop protocol independent interfaces that are complementary to realtime systems. They plan to use the work done by SEI as an Ada binding to the SAFENET Lightweight protocol suite as a base document for their work.

In light of the nature of the 1003.21 work as well as the P1003.12, 1238, 1238.1 and 1224.2 work and their close association with the Network and Communications Interfaces service class, the OSSWG needs to monitor progress in these groups closely.

In a system using components based on NGCR standards, there will frequently be a hierarchy of networked communication, data storage, and processing functions. At the base of this hierarchy may be a number of processing or storage units on a single board connected by an onboard bus. At the next level will be FUTUREBUS+ or non-NGCR backplane busses (e.g., VME). At the next level there may be SAFENET, MIL-STD-1553B data busses, or non-NGCR-defined LANs. At the highest level, but outside the scope of this set of requirements, there may be communications among systems on different Navy platforms. In some application domains and for some application functions, the OSIF must provide explicit access to networked communication, data storage, and processing functions for both NGCR-defined communication components and similar non-NGCR-defined components. This is in addition to the use of these capabilities implied in many other requirements. Two processes make up a communications transaction regardless of their location. This includes either two processes across a communications link or two processes residing on the same processor.

OSSWG has expressed some concern that requirements 8.3 (Acknowledged Connection-Oriented Service) and 8.4 (Unacknowledged Connection-Oriented Service) actually dictate two protocol-specific implementations of Connection-Oriented Service intended to exploit a trade-off between highly reliable delivery and high performance. The same can be said of requirements 8.5 (Acknowledged Datagram Service) and 8.6 (Datagram Service). It has been suggested that these requirements be re-worded to state true requirements; that is, Connection-Oriented Service or Datagram Service with specified levels of reliability and performance. This is more in keeping with the P1003.12 concept of Quality of Service parameters, and isolates the requirements from dependency on current network protocol implementations. However, since the Acknowledged/Unacknowledged paradigm is so pervasive throughout current networking technology, OSSWG is reluctant to change these requirements without further study. OSSWG recommends that this issue be addressed as part of an overall review of all the OCD requirements.

3.8.1 Interface to NAVY Standard Network

This unfulfilled requirement is classified as "a" (essential).

This requirement is partially covered by the work of 1238 and 1238.1 which provide interfaces to the SAFENET OSI suite; 1224.2 which provides an interface to directory services; and P1003.12 and 1003.21 which provide interfaces to the SAFENET lightweight suite. 1003.21 is attempting to make their interface also applicable to backplane buses such as Futurebus+. Only 1003.21 plans to provide an Ada binding to its interface. Additionally, the POSIX 12XX series of standards does not currently include interfaces for ROSE or network management which are needed to support the SAFENET OSI suite.

Requirement: The operating system shall provide explicit interfaces to and control of NGCR standard communications implementations. These implementations shall include but not be limited to implementations of Futurebus+ and SAFENET.

Description of Delta: POSIX/12XX provide no interfaces for ROSE or network management, both of which are needed to provide a complete interface to SAFENET.

Recommendation: Pursue/support PARs for interfaces to ROSE and network management.

3.8.2 Interfaces to Other Network and Communication Entities

This requirement is met in various ways. Explicit interfaces exist in P1003.12 for interfacing to networks (Ethernet and FDDI), usually using additional protocols (TCP/IP and ISO). Interfaces also exist to access devices via 1003.1 Input and Output Primitives. Although device drivers are needed to access devices such as MIL-STD-1553B, these interfaces can be used in a portable manner. Finally, it is generally accepted that access to backplane busses (VME, MULTIBUS, and Pi-Bus) is not explicitly given to applications and the details of backplane communication are regarded as an implementation issue.

3.8.3 Acknowledged Connection-Oriented Service

There is no delta for this requirement. The work of the 1351, 1003.12, and 1003.21 groups will satisfy this requirement. Only the 1003.21 group plans to provide an Ada binding.

There is some concern within the OSSWG that 1003.21 will "overfulfill" this requirement; that is, if 1003.21 develops alternative interfaces to Acknowledged Connection-Oriented Service (as opposed to simply Ada bindings for those already developed by the 1238 and 1003.12 groups), it is not clear that having two sets of interfaces will be advantageous.

3.8.8 Broadcast/Multicast Service

This unfulfilled requirement is classified as "a" (essential).

There will be no delta for this requirement if the 1003.21 work proceeds as planned. Broadcast and multicast requirements appear in the 1003.21 requirements document and broadcast and multicast services appear in the 1003.21 Ada binding base document.

Requirement: The OSIF shall provide for the selection of broadcast/multicast communication services.

Description of Delta: No delta/dependent on 1003.21 group work.

Recommendation: Monitor/influence 1003.21 group work. OSSWG is concerned that the 1003.21 group is not making adequate progress in defining this interface and may not provide a C-Language binding to this interface; therefore the 1003.12 group should be considered as a backup.

3.8.9 K-Acknowledged Multicast Service

This unfulfilled requirement is classified as "a" (essential).

There will be no delta for this requirement if the 1003.21 work proceeds as planned. The 1003.21 requirements document discusses a k-acknowledged multicast service and the 1003.21 Ada binding base document specifies an active-group-integrity quality-of-service parameter on multicast services which also implies k-acknowledgment.

Requirement: The OSIF shall provide for the selection of multicast communication services that ensure reliable delivery to at least k of n multicast group members.

Description of Delta: No delta/dependent on 1003.21 group work.

Recommendation: Monitor/influence 1003.21 group work. OSSWG is concerned that the 1003.21 group is not making adequate progress in defining this interface and may not provide a C-Language binding to this interface; therefore the 1003.12 group should be considered as a backup.

3.8.10 Atomic Multicast Service

This unfulfilled requirement is classified as "a" (essential).

There will be no delta for this requirement if the 1003.21 work proceeds as planned. A multicast transaction requirement appears in the 1003.21 requirements document and a multicast transaction service appears in the 1003.21 Ada binding base document.

Requirement: The OSIF shall provide for the selection of reliable, atomic multicast communications services.

Description of Delta: No delta/dependent on 1003.21 group work.

Recommendation: Monitor/influence 1003.21 group work. OSSWG is concerned that the 1003.21 group is not making adequate progress in defining this interface and may not provide a C-Language binding to this interface; therefore the 1003.12 group should be considered as a backup.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
8.1	Partially	Modification	a
8.2	Yes	None	-
8.3	Yes	None	-
8.4	Yes	None	-
8.5	Yes	None	-
8.6	Yes	None	-
8.7	Probably	Insertion	a
8.8	Probably	Insertion	a
8.9	Probably	Insertion	a
8.10	Probably	Insertion	a

3.9 PROCESS MANAGEMENT INTERFACES

In general, the POSIX Standards support Service Class 9, Process Management, in a substantially complete way for both the Pthread Model and the POSIX process model.

OSSWG requires a single unit of concurrency, namely the "process." 1003.1 and 1003.4 support this requirement via the POSIX process model, while P1003.4a adds a second level of concurrency (within a POSIX process) called POSIX threads. Depending on the application, an OSSWG "process" may be either a POSIX process or a POSIX thread. Furthermore, some applications (particularly in Ada) may require simultaneous use of both concurrency models. Therefore, this analysis separately considers each requirement as it is met by POSIX processes and by POSIX threads (Pthreads).

The ability to create processes is an essential part of the POSIX interface and an Ada binding to POSIX without processes would be incomplete. Nevertheless, it is possible that the POSIX process model is at odds with the Ada multitasking model, particularly since a standard mapping between these two models does not exist. Therefore, Ada programmers should be aware of potential conflicts that can occur when creating POSIX processes.

In an attempt to reconcile the Ada and POSIX models of concurrency there seems to be three potential mappings: 1) each Ada task is a POSIX process, 2) each Ada program is a POSIX process, or 3) there is not a simple relationship between POSIX processes and Ada tasks. The choice that causes the least conflict between Ada and POSIX is to require that the POSIX Ada standard interface to POSIX impose a virtual one-to-one correspondence between processes and program executions. That is, an Ada program execution should act, feel, and look as if it is running as a single POSIX process. This equivalence between a POSIX process and an Ada program means that one cannot differentiate between the two POSIX calls. This choice has the virtue of raising the fewest problems and resolving many issues cleanly. The P1003.5 standard accommodates this idea by isolating those features of POSIX that deal with process creation within the packages POSIX_Process_Primitives, POSIX-Unsafe_Process_Primitives, and POSIX_Process_Identification.

3.9.1 Create Process

The requirement for Create Process (9.1) is directly met for Pthreads by P1003.4a plus the interprocess communication facilities of 1003.4. The Create Process (9.1) requirement is met for processes by the fork and exec interfaces of 1003.1, the spawn interface of P1003.4b, the scheduling interface of 1003.4, plus the communication and synchronization interfaces of 1003.4. The use of these interfaces in combination to meet the requirement is adequate since the requirement is stated as "shall support." P1003.1a provides a system() interface to 1003.2 shell commands to meet this requirement.

3.9.2 Terminate Process

This unfulfilled requirement is classified as "a" (essential).

The requirement for Terminate Process (9.2) is almost met for Pthreads by P1003.4a plus the interprocess communication facilities of 1003.4. The requirement for Terminate Process (9.2) is directly met for POSIX processes by 1003.1 process interfaces plus 1003.4 process attributes and interprocess communication facilities. Also, for processes only, 1003.2 provides the "kill" shell command to meet this requirement.

Requirement: The OSIF shall support the ability to terminate a process and recover all resources associated with that process.

Description of Delta: Pthread_kill() cannot unconditionally terminate another thread (it will terminate the entire process instead), and pthread_cancel() also cannot unconditionally terminate another thread (that thread may have disabled cancellation). Therefore, there is no interface to unconditionally terminate another thread.

Recommendation: OSSWG should influence the 1003.4 working group to provide an interface by which one thread may unconditionally terminate another thread. This appears to be a technical correction (or addition) to P1003.4a, and can possibly be achieved through the P1003.4d project.

3.9.3 Start Process

The requirement for Start Process (9.3) was purposely rejected as a separate interface by P1003.4a in favor of use of the Pthread synchronization primitives to achieve the same effect whenever process creation and startup must be separately managed. This alternative capability is adequate to meet this "shall support" requirement. The requirement for Start Process (9.3) is also not separately addressed for POSIX processes. The requirement is met by the 1003.1 fork(), execl(), and execve() interfaces and by the P1003.1a system() interface to 1003.2 shell commands. It is indirectly supported via the 1003.1 and 1003.4 process synchronization interfaces, much as in the case of Pthreads. Since this is a "shall support" requirement, it is met by a combination of POSIX process synchronization primitives.

3.9.4 Stop Process

The requirement for Stop Process (9.4) is not addressed by POSIX for either Pthreads or POSIX processes. The whole concept of stopping a process for subsequent restart (from a point other than where it was stopped) is considered by POSIX as an application dependent variant of a thread or process becoming blocked and subsequently unblocked. Since POSIX does indirectly support Suspend Process (q.v.), and standard languages support both local and non-local jumps, this "shall support" requirement is considered met by POSIX.

3.9.5 Suspend Process

The requirement for Suspend Process (9.5) is met for both Pthreads and POSIX processes by combinations of interfaces in 1003.1, 1003.4, and P1003.4a. Although no Pthread or POSIX process interface explicitly provides each of these capabilities, the requirement is met by combining interfaces. The POSIX community regards asynchronously affecting the state of another process or thread as a dangerous capability, and suggests that this be accomplished by asynchronously or synchronously requesting the other thread change its own state.

3.9.6 Resume Process

The requirement for Resume Process (9.6) is met for both Pthreads and POSIX processes by combinations of interfaces in 1003.1, 1003.4, and P1003.4a. Although no Pthread or POSIX process interface explicitly provides each of these capabilities, the requirement is met by combining interfaces. The POSIX community regards asynchronously affecting the state of another process or thread as a dangerous capability, and suggests that this be accomplished by asynchronously or synchronously requesting the other thread change its own state.

3.9.7 Delay process

The requirement for Delay Process (9.7) is met for both Pthreads and POSIX processes by combinations of interfaces in 1003.1, 1003.4, and P1003.4a. Although no Pthread or POSIX process interface explicitly provides each of these capabilities, the requirement is met by combining interfaces. The POSIX community regards asynchronously affecting the state of another process or thread as a dangerous capability, and suggests that this be accomplished by asynchronously or synchronously requesting the other thread change its own state. Also, "delay until" semantics, although not directly supported for POSIX processes or Pthreads, can be achieved through a combination of the 1003.4 clocks and timers interfaces and 1003.1, 1003.4 and P1003.4a signal interfaces. 1003.2 provides the "sleep" shell command to meet this requirement.

3.9.8 Interprocess Communication

The requirement for Interprocess Communication (9.8) is directly met for Pthreads by P1003.4a plus the interprocess communication facilities of 1003.4. The requirement for Interprocess Communication (9.8) is directly met for POSIX processes by 1003.1 process interfaces, P1003.4 Synchronization plus 1003.4 process attributes and interprocess communication facilities. P1003.12 explicitly provides interprocess communication interfaces for a distributed/networked environment.

3.9.9 Examine Process Attributes

The requirement for Examine Process Attributes (9.9) is directly met for Pthreads by P1003.4a , Execution Time Monitoring of P1003.4b, plus the interprocess communication facilities of 1003.4. The requirement Examine Process Attributes (9.9) is directly met for POSIX processes by 1003.1 process interfaces, Execution Time Monitoring of P1003.4b, plus 1003.4 process attributes and interprocess communication facilities. 1003.2 provides the "ps" shell command to meet this requirement.

3.9.10 Modify Process Attributes

The requirement for Modify Process Attributes (9.10) is directly met for Pthreads by P1003.4a , Execution Scheduling of P1003.4b plus the interprocess communication facilities of 1003.4. The requirement for Modify Process Attributes(9.10) is directly met for POSIX processes by 1003.1 process interfaces plus 1003.4 process attributes and interprocess communication facilities.

3.9.11 Examine Process Status

This unfulfilled requirement is classified as "a" (essential).

The requirement for Examine Process Status (9.11) is not adequately covered either for Pthreads or POSIX processes. Interfaces to enable one Pthread or POSIX process to obtain the current status of another must be added.

Examine POSIX Process Status

Requirement: The OSIF shall provide the ability for processes to examine the current status of a particular process. Note that status here is not intended to include cumulative execution time; the capability to obtain cumulative execution time is covered as requirement 3 in service class 13 (synchronization and scheduling).

Description of Delta: The wait() and waitpid() functions provide limited status (terminated, stopped, and why (e.g., caused by which signal)) on limited processes (child processes). Richer status information is required. The ability to examine status of general processes (i.e., non-children) is required. 1003.2 provides the "ps" command, but no API (system call version) is provided.

Resolution Alternatives:

1. Enhance existing 1003.1 wait() and waitpid() interfaces to include this capability. Extensions of wait() and waitpid() to provide richer status information and to allow status querying to general processes are discussed in 1003.1 but are not included in the standard. It is unlikely that a consensus to include the extensions could be achieved.

2. Incorporate an API to 1003.2 "ps" command functionality into a POSIX standard. The functionality should be incorporated as a system call and also as a command ("ps" is available only as a command in 1003.2).

Recommendation: The P1003.7 drafts should be reviewed to determine whether a system call version of "ps" is on the agenda. The 1003.7 group should be approached with a proposal to include the capability for examining process status in one of their drafts if this is not already on the agenda.

Examine POSIX Thread Status

Requirement: The OSIF shall provide the ability for threads to examine the current status of a particular thread. Note that status here is not intended to include cumulative execution time; the capability to obtain cumulative execution time is covered as requirement 3 in service class 13 (synchronization and scheduling). Note also that this requirement has particular relevance for Ada applications, as specified in paragraph 3.16.10. Changes to the recommendations should take that fact into account.

Description of Delta: The pthread_join function provides limited status information: whether a thread has terminated. Richer status information is required.

Resolution Alternatives:

1. Investigate extending 1003.2 "ps" command functionality to threads and incorporating a system call version into a POSIX standard. Although threads are addressed in the 1003.4 working group, that group does not consider such an interface appropriate to standardize at this time due to lack of existing practice and its lack of relevance to the realtime charter. The 1003.7 group seems to be the likely place to address this in conjunction with the API for process status discussed above.

Recommendation: Alternative 1 should be pursued in the 1003.7 working group (for a thread status API). 1003.2 should be requested to add a thread status command (possibly based on this API at a later date), but this is less crucial to fulfilling the OSSWG requirement.

3.9.12 Process (Thread) Identification

The requirement for Process Identification (9.12) is directly met for Pthreads by P1003.4a plus the interprocess communication facilities of 1003.4 and for POSIX processes by 1003.1 process interfaces, 1003.4 process attributes and interprocess communication facilities, and Process Management interfaces of P1003.4b. 1003.2 provides the "ps" shell command to meet this requirement.

3.9.13 Save/Restart Process

This unfulfilled requirement is classified as "a" (essential).

The requirement for Save/Restart Process (9.13) is directly met for POSIX processes by the P1003.1a Process Checkpoint and Restart capability. This requirement is not met for Pthreads, however, since P1003.4a defines no equivalent per-thread capability. This is understandable since this P1003.4a capability is relatively new.

Requirement: The OSIF shall support the ability for processes to be restarted from a saved state. Note that this requirement has particular relevance for Ada applications, as specified in paragraph 3.16.6. Changes to the recommendations should take that fact into account.

Description of Delta: At this time, these interfaces are not provided for Pthreads.

Resolution Alternatives:

1. Investigate checkpointing/restarting of threads, possibly in the context of a broader OSSWG fault tolerance proposal. Consider 1003.7 as forums for making proposals.

2. Levy the requirements and the OSIF general requirements on vendors but do not provide a standard as such. This alternative relies on vendors to develop some commercial existing practice in this area on which to potentially standardize at a later date.

Recommendation: Alternative 1 is recommended, while it is recognized that program managers can always resort to alternative 2. Checkpointing a thread that is sharing memory with other threads seems to be difficult and demands further study.

3.9.14 Program Management Function

The requirement for Program Management (9.14) is directly met for Pthreads by P1003.4a plus the interprocess communication facilities of 1003.4. The requirement for Program Management (9.14) is directly met for POSIX processes by 1003.1 process interfaces plus 1003.4 process attributes and interprocess communication facilities.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
9.1	Yes	None	-
9.2 (Process)	Yes	None	-
9.2 (Pthread)	Partially	Modification	a
9.3	Yes	None	-
9.4	Yes	None	-
9.5	Yes	None	-
9.6	Yes	None	-
9.7	Yes	None	-
9.8	Yes	None	-
9.9	Yes	None	-
9.10	Yes	None	-
9.11	No	Insertion	a
9.12	Yes	None	-
9.13 (Process)	Yes	None	-
9.13 (Pthread)	No	Insertion	a
9.14	Yes	None	-

3.10 PROJECT SUPPORT ENVIRONMENT INTERFACES

Two "profile" related architectures are possible for the implementation of the OSSWG requirements for debug support (see OCD Appendix, 20.10.1) and execution history (OCD Appendix, 20.10.2).

In architecture A, the process being debugged interfaces to the debugger, which in turn interfaces to the operating system. Conceptually, this is the equivalent to the debugged process executing in an application debugger "shell" that interfaces to the supplied operating system. (Note: This architecture appears to be the one assumed by earlier versions of the OSSWG Delta Document.) Alternatively, it can be thought of as the capability to create an instrumented, self-monitoring copy of the target process. This architecture has the following characteristics:

1. It is most naturally applied to general-purpose RAM-based development systems. These systems would support compiling, linking, etc.

2. There is an essential link between the debugger and other process development tools (i.e., the compilers, linkers, etc.). The debug capability accesses the process at the source level.

3. The debugger is assumed to reside upon the application platform.

4. The debug functionality is supplied at the application level and not the operating system level. Execution history can also naturally be maintained at this application level without additional OS functionality.

5. There is currently (for a given language) a body of practice in place that supports the Requirements Document with an indirect "virtual" debug capability (if not the direct "physical" capability, i.e., the direct alteration of the registers of an executing process).

Given the above characteristics, there does not appear to be any delta at the "kernel" POSIX level. Because of the strong relationship between the debugger and the compiler, there might be some language (Ada, C, etc.) binding considerations. This would probably be a direct binding between the language and the debugger (i.e., tool to tool) not involving the OS.

In architecture B, the debugger interfaces to the operating system, which in turn interfaces to the process being debugged. (Note: This is the architecture that appears to be implied by figure 10.2-2 of the OCD.) Conceptually, this can be viewed as supplying external access to a "target" system via operating system services. This architecture has the following characteristics:

1. It is most naturally applied to special-purpose PROM/EPROM-based systems (e.g., flight control computers).

2. There is not necessarily a link between the debugger and the compiler, linker, etc., of the target process. The debug capability accesses the system at the code level.

3. There is, in general, a physical/logical separation between the application platform and the POSIX Standard Environment (PSE) host platform. A communication protocol may be necessary as part of the debugger/OS interface.

4. The debug functionality would be supplied by the application platform OS but not necessarily by the Application Program Interface (API.) Execution history would also be maintained within the OS.

5. There is little standard practice with respect to this architecture. It is, in general, dependent on the implementation of the test bed hardware.

Given the above characteristics, the current POSIX primitives for process control do not give the degree of control needed to support the debug requirements. It would be difficult to "single-step" a process with the current services. In addition, full debug control may require the capability to override normal operating system functions (i.e., scheduling). It may be required to "idle" a target system so that it can be "patched." Such actions have an "anti-operating system" viewpoint. New POSIX services (with syntax, semantics, and protocols) would need to be provided to satisfy the OCD requirements. However, such services would need to be privileged and not part of the basic API available to every application. Execution history would need to be added to the OS functionality. Note that in some systems debug services are part of the operating system (and are removed in the operational system). They may only be recording debug information that the application accesses runtime. In that case, interfaces such as the POSIX read-file (paragraph 6.4, 1003.1 and paragraph 6.1, 1003.5) may be adequate.

Based on the above discussion, the debug requirement would currently be supported by POSIX for a number of profiles (although a considerable effort in generating a debug application would also be necessary) and not supported by POSIX for other profiles.

3.10.1 Debug Support

This unfulfilled requirement is classified as "c" (may be deferred).

<u>Requirement:</u> The OSIF shall support the debugging of applications, specifically supporting the following capabilities:

1. Examine registers
2. Alter registers
3. Set/clear breakpoint
4. Set/clear watchpoint
5. Single step execution
6. Continue execution
7. Examine memory
8. Alter memory
9. Query process environment
10. Query call stack

Description of Delta: Depending on the architecture, there is either no delta or a considerable delta. POSIX standards do not directly address application debugging. However, vendors who are marketing POSIX-compliant systems are certain to include debugging support for application developers as part of their system. POSIX standards should contain debug support to ensure that a common set of debug capabilities exists across different POSIX-compliant systems. At present, it is unclear where debug support should be included in the POSIX standards.

Resolution Alternatives:

1. Redefine the requirement so that it is limited to application platform resident debug tools. This would eliminate the delta. PSESWG would be responsible for standardization of the resulting debug interface (tool-to-tool, tool-to-OS, etc.). This seems contrary to the intent of the requirement in section 4.1.10 of the OCD.

2. Insert new service primitives into the POSIX standard. Because there is no standard practice to support these primitives, both the syntax and semantics for them (in terms of the UNIX/C environment or the Ada tasking model) would also have to be determined. This alternative does not fit the NGCR methodology of building on current practice.

3. Declare that the OS/PSE interface is not done through the API and thus is not part of the MIL-STD-OSIF. Again, PSESWG would be responsible for defining and standardizing an appropriate OS/PSE interface including potential communication protocols.

4. Wait for the OSSWG/PSESWG boundary paper to determine the scope of the problem.

Recommendation: OSSWG recommends alternatives 3 and 4.

3.10.2 Execution History

This unfulfilled requirement is classified as "c" (may be deferred).

Requirement: The OSIF shall support the ability to monitor the execution history of a process, including such information as

1. Frequency of calls
2. Length of calls
3. Missed deadlines
4. Length of queues
5. Tasking of runtime systems
6. Dynamic paging activity
7. Memory allocation
8. What OS services being used

Description of Delta: An interface to support the collection and reporting of execution statistics of a process is not addressed in the POSIX standards. Execution statistics are needed to evaluate and tune process and system performance. 1003.1 would be a logical place to incorporate an interface for the collection and reporting of execution statistics of a process.

An application platform resident debug program could easily implement this requirement within a debug application "shell." Even if no debug application is assumed, most of these statistics could be achieved using POSIX service primitives within an application (except for missed deadlines). This execution history functionality would become the responsibility of the application layer and not readily available to an external PSE. A cleaner solution would be to enhance the "ps" command to include some history status as part of its functionality.

Resolution Alternatives:

1. Redefine the requirement so that it is limited to application platform resident PSE tools. This would eliminate the delta. PSESWG would be responsible for standardization of the resulting execution history tool interface.

2. Modify the current status service primitives in the POSIX standard to include history information. This would make history information more readily available to both an application and an external PSE.

3. Wait for the OSSWG/PSESWG boundary paper to determine the scope of the problem.

Recommendation: OSSWG recommends alternatives 2 and 3.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
10.1	No	Insertion	c
10.2	No	Insertion	c

3.11 RELIABILITY, ADAPTABILITY, AND MAINTAINABILITY INTERFACES

In general, the POSIX standards support service class 11 in a rudimentary way. There are two areas that are not complete:

1. Basically POSIX provides reactive fault management, while OSSWG requires proactive behavior. Attempting to support proactive requirements on top of a reactive interface will result in performance penalties. The existing (proactive) services are highly-oriented toward providing event services (via the "signal" concept), while downplaying fault reportage.

2. POSIX does not provide adequate monitoring, coordination, and recording services.

For the purposes of this subsection, it is important to differentiate modules of the operating system itself from modules that do "generalized input/output." The latter are often called "device drivers." In the latter case, it is fairly straightforward for an application to provide all the services specified by OSSWG. For instance, an interface can be added to set a fault threshold for retrying a message transmitted via a UHF radio. Since the provided functionality is under direct control of the application and is not required of the general operating system (i.e., POSIX), the potential functionality of application-developed generalized I/O modules will not be further considered.

Refer to the Executive Summary in section 3.5 (Event and Error Interfaces) for additional information pertinent to this section.

3.11.1 Fault Information Collection

This unfulfilled requirement is classified as "a" (essential).

OSSWG requirement Fault Information Collection (11.1) is partially covered by POSIX. While the event interfaces exist and error interfaces are provided for individual processes, there are no fault coordination or distribution interfaces. Furthermore, an event ("signal" in POSIX) can be blocked without the sender's knowledge or any other reportage.

Requirement: The OSIF shall provide for specifying the collection of available fault information.

Description of Delta: This requirement refers to specifying the collection of fault information coming into the OS across the OSIF for subsequent distribution according to requirement 11.2. POSIX says nothing about such fault information collection.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.2 Fault Information Request

This unfulfilled requirement is classified as "a" (essential).

OSSWG requirement Fault Information Request (11.2) is partially covered by POSIX. While the event interfaces exist and error interfaces are provided for individual processes, there are no fault coordination or distribution interfaces. Furthermore, an event ("signal" in POSIX) can be blocked without the sender's knowledge or any other reportage.

Refer to section 3.5.2 (Event and Error Distribution) for additional information related to Fault Information Request.

Requirement: The OSIF shall provide for the receipt of fault information on request.

Description of Delta: POSIX provides for the distribution of errors to the requesters of individual functions. Each function specifies which errors all POSIX implementations must detect and which are optional. Paragraph 2.4 of 1003.1 lists the possible errors. However, "implementations may support additional errors not included in this clause, may generate errors included in this clause under circumstances other than those described in this clause, or may contain extensions or limitations that prevent some errors from occurring" (paragraph 2.4, 1003.1). "If more than one error occurs in processing a function call, this part of ISO/IEC 9945 does not define in what order the errors are detected; therefore, any one of the possible errors may be returned" (paragraph 2.4, 1003.1).

The OSIF requires that all possible fault information be available, not just one of the errors that occurred. It also requires that there be a means for coordinating the distribution of fault information, as for example to a single process responsible for fault analysis.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.3 Diagnostic Tests Request

This unfulfilled requirement is classified as "a" (essential).

This requirement is not supported by POSIX.

Requirement: The OSIF shall provide for the initiation of diagnostic tests on specific request. The OSIF shall support initiation of diagnostic tests at specified intervals. This is a necessary OSIF requirement.

Description of Delta: POSIX does not provide for the initiation of diagnostic requests.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.4 Diagnostic Tests Results

This unfulfilled requirement is classified as "a" (essential).

This requirement is not supported by POSIX.

Requirement: The OSIF shall provide the ability to determine the results of diagnostic tests.

Description of Delta: POSIX does not provide for determining the results of diagnostic tests.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.5 Operational Status

This unfulfilled requirement is classified as "a" (essential).

This requirement is barely supported by POSIX.

Requirement: The OSIF shall provide access to the operational status of all system components.

Description of Delta: POSIX essentially does not provide access to the status of system components. POSIX does inform a requester of the success or failure of a requested function from which the requester may derive some status information. Specifically, [ENXIO], no such device or address, and [EIO], input/output error, are possible error returns (paragraph 2.4, 1003.1). However, in the case of [EIO], "any other error-causing operation on the same file descriptor may cause the [EIO] error indication to be lost" (paragraph 2.4, 1003.1).

Process termination status is available to an application that has issued a 1003.1 wait() for a child process termination.

Also, thread termination "makes the value status available to any successful join with the terminating thread" (P1003.4a).

Some systems, however, may maintain operational status in a file. In that case interfaces such as the POSIX read-file (paragraph 6.4, 1003.1 and paragraph 6.1, 1003.5) may be adequate to obtain this information.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.6 Fault Detection Thresholds

This unfulfilled requirement is classified as "a" (essential).

An application can choose to retry an operation, as specified by requirement 11.6, but retries are risky since the state of the operating system is not well-defined subsequent to an error. Furthermore, no other part of requirement 11.6 (fault detection thresholds), such as classifying the component as suspect, is provided.

Requirement: The OSIF shall provide for specifying fault detection thresholds, which shall include, but not be limited to, the following:

1. Number of retry attempts, if applicable, that shall be made before an error is determined to be a non recoverable fault.

2. Maximum number of correctable errors that, if detected within a specified time, will c[illegible]y the component as suspect or treat the collective errors as a non recoverable fault.

Description of Delta: Within the limits discussed under requirement 5.2 - i.e., POSIX does not provide for coordination in the distribution of events and errors - some user-selectable error processing alternatives are available. Processes can mask signals (paragraph 3.3.1.2, 1003.1). Processes can also choose among three types of actions that they can associate with a signal: a default action, ignore, and a signal catching function (paragraph 3.3.1.3, 1003.1). Retries and accumulation of occurrences would then be the responsibility of the individual processes. In particular, occurrences of a particular event or error could not be collected for several processes or for the system as a whole through the interface. This discussion also applies to threads as per P1003.4a signal handling.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.7 Fault Isolation

This unfulfilled requirement is classified as "a" (essential).

This requirement is barely supported by POSIX.

Requirement: The OSIF shall support the isolation of faults to a particular component.

Description of Delta: POSIX provides little support for the isolation of faults, either in the sense of precisely determining the component causing the fault or in the sense of containing the fault to prevent it from damaging the rest of the system, which assumes determining the source of the fault.

Using error numbers from failed function calls to determine the responsible component is unsatisfactory because "if more than one error occurs in processing a function call, this part of ISO/IEC 9945 does not define in what order the errors are detected; therefore, any one of the possible errors may be returned" (paragraph 2.4, 1003.1). Furthermore, error numbers do not provide enough information as to the nature of the error. For instance, POSIX may return [ENXIO] when a device does not exist, a request was made beyond the limits of the device, or a tape drive is not online or a disk pack is not loaded on a drive (paragraph 2.4, 1003.1). A prerequisite to fulfilling this requirement is to also fulfill requirements 11.3 and 11.4 to determine faulty components and requirement 11.10 to prevent a faulty component from causing further damage.

Device Control (P1003.4b) may permit device fault isolation, but is not required to do so.

If requirement 5.1 is fully satisfied, mechanisms will be available to support fault isolation.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.8 Fault Response

This unfulfilled requirement is classified as "a" (essential).

This requirement is barely supported by POSIX.

Requirement: The OSIF shall specify the actions to be taken on the occurrence of a fault. The OSIF shall support (at least) the following actions:

1. Restart at a specified point for a specified fault.
2. Use of specified components as backup for faulty components.
3. Stop when a specified minimum set of components is no longer available.
4. Schedule of a specified process.
5. Report to another node.

Description of Delta: Within the limits discussed under requirement 5.2 - i.e., POSIX does not provide for coordination in the distribution of events and errors - some user-selectable error processing alternatives are available. Processes can mask signals (paragraph 3.3.1.2, 1003.1). Processes can also choose between three types of actions that they can associate with a signal: a default action, ignore, and a signal catching function (paragraph 3.3.1.3, 1003.1). Restart, stop (provided requirement 11.5 is fulfilled), schedule, and report actions would then be the responsibility of the individual processes. Directing the use of specific hardware components is not a function of POSIX. Consistent handling of a particular fault would not be a function of the interface but would have to be a design convention for each system. This discussion also applies to threads as per P1003.4a signal handling.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.9 Reconfiguration

This unfulfilled requirement is classified as "a" (essential).

This requirement is barely supported by POSIX.

Requirement: The OSIF shall support the dynamic reconfiguration of hardware and software.

Description of Delta: POSIX does not support reconfiguration of hardware and does not explicitly support reconfiguration of software. POSIX does provide ways to create and terminate processes. 1003.1 allows processes to spawn and execute child processes and to effect normal and abnormal termination of processes . P1003.4a expands this capability to also allow for the creation, termination, and cancellation of threads, though currently a thread cannot be unconditionally terminated by another thread. Thus, a mechanism external to the OS and, therefore, not included as such in the OSIF, such as an overall "parent" process or processes responsible for software configuration, could answer the software reconfiguration part of this requirement. Again, because POSIX does not provide for the centralization of such functions within a system, effecting software reconfiguration in this manner may require extensive management and coordination, particularly between processes, during system development and be unique to each system developed.

Some systems may only require a more rudimentary form of reconfiguration whereby the new configuration is recorded in a file. Then, either the operating system monitors the file and effects the reconfiguration and/or the application directs a reboot of the system which effects the reconfiguration. In such a case reconfiguration, as far as the application is concerned, can be realized through interfaces such as the POSIX write-file (paragraph 6.4, 1003.1 and paragraph 6.1, 1003.5).

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.10 Enable/Disable System Component

This unfulfilled requirement is classified as "a" (essential).

POSIX coverage of requirement 11.10 (Enable/Disable System Component) is also unacceptably poor, even though it does provide some of the functionality demanded by OSSWG. In particular, POSIX permits a component to be terminated if (1) the unit to be terminated is a software "process," and (2) the process correctly receives and handles a "signal kill."

Requirement: The OSIF shall provide the ability to enable or disable a specified system component on request.

Description of Delta: POSIX does not provide the ability to enable or disable hardware components, although I/O work in 1003.7 and/or the Device Control interface in P1003.4b may apply. POSIX does provide ways to create and terminate processes. 1003.1 allows processes to spawn and execute child processes (paragraph 3.1) and to effect normal and abnormal termination of processes (paragraphs 3.2 and 3.3). P1003.4a expands this capability to also allow for the creation, termination, and cancellation of threads, though currently a thread cannot be unconditionally terminated by another thread.

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.11 Performance Monitoring

This unfulfilled requirement is classified as "a" (essential).

A few performance statistics are available from POSIX. For instance, a process can measure its CPU time and some information about its file utilization. But otherwise POSIX does not meet the performance monitoring requirement, 11.11.

Requirement: The OSIF shall support queries for snapshots of resource utilization and enabling or disabling monitoring of each resource.

Description of Delta: POSIX provides limited support for obtaining snapshots. 1003.1 provides for obtaining process and child process execution and system CPU times and 1003.4b provides interfaces for obtaining execution times of an arbitrary process or thread. 1003.1 also provides for obtaining file information including time of the last access, time of the last data modification, and time of the last file status change .

Recommendation: OSSWG recommends monitoring and participating in related standards efforts at UNIX International; Open Software Foundation; POSIX Services for Reliable, Available, and Serviceable

Systems group; and X3T8. When these groups develop mature standards, move appropriate interfaces into POSIX.

3.11.12 Set Resource Utilization Limits

This requirement is directly met by P1003.1a Resource Limits, the numerical limits defined by 1003.1 and its amending documents, and the Sporadic Server and CPU Time Clocks of P1003.4b.

3.11.13 Resource Utilization Limits Violation

This requirement is directly met by P1003.1a Resource Limits and the error returns in 1003.1 and its amending documents which indicate that one of the numerical limits has been exceeded.

3.11.14 Checkpoint Data Structures

Requirement Checkpoint Data Structure (11.14) is completely met by P1003.1a Checkpoint a Process or Set of Processes along with Restart Execution of a Process or Process Family.

It should be noted that the Checkpoint function saves all the process state information necessary to restart a process or family of processes. Particularly if a system needs to checkpoint only data structures or only certain data structures, other interfaces to consider are the Memory Mapping interfaces in 1003.4 and P1003.20. Memory Mapping allows an application to establish a mapping between a part of the process address space and a memory object such as a file on a storage medium. If the application chooses a map-shared option for use with this interface, write references to the specified address space will also change the file on the storage medium. Alternatively the application may request a Synchronize function at its own discretion which updates the file to agree with the specified address space.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
11.1	Partially	Insertion	a
11.2	Partially	Insertion	a
11.3	No	Insertion	a
11.4	No	Insertion	a
11.5	No	Insertion	a
11.6	No	Insertion	a
11.7	No	Insertion	a
11.8	No	Insertion	a
11.9	No	Insertion	a
11.10	No	Insertion	a
11.11	No	Insertion	a
11.12	Yes	None	-
11.13	Yes	None	-
11.14	Yes	None	-

3.12 RESOURCE MANAGEMENT INTERFACES

This service class is partially supported by 1003.1, 1003.4, and P1003.7.

3.12.1 Virtual Memory Support

This unfulfilled requirement is classified as "a" (essential).

Requirement: The OSIF shall support the selection of the virtual memory utilization parameters.

Description of Delta: This requirement refers to controlling virtual memory utilization such as the paging algorithm. POSIX P1003.4b provides an Advisory Information interface, madvise(), which advises the operating system of the application's expected memory access behavior. However, this information is purely advisory, and may not provide sufficient control over virtual memory parameters for some realtime applications.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. There has historically been much opposition within POSIX to the inclusion of interfaces that place requirements on the underlying architecture. Opponents argue that applications that presume a particular method of memory management will not be portable to all architectures. Vendors who do not support virtual memory architectures would be undesirably forced to provide such a function. The requirement for such an interface might also inhibit the development of new and better methods of memory management. Typically, UNIX operating systems from vendors that support virtual memory, do provide limited control over the use of virtual memory. The HP-UX chatr() command is a good example. A complete virtual memory support interface would best be added to P1003.7. Even though 1003.2 might also be a logical place for such an interface, OSSWG has chosen to avoid inclusion of 1003.2 in the OSIF primarily for performance reasons.

2. Assume a standard outside POSIX. Often, the link editor has options that allow for some control over a process's use of virtual memory. The C or Ada standard might include options to allow selection of virtual memory characteristics. These would be embedded in the executable header information similar to the link editor in HP-UX.

3. Develop a new military standard. This is a less acceptable alternative than 1 because it is external to the OSIF baseline. It is suggested that any new military standard be based on de facto UNIX or industry standard(s), if any exist.

Recommendation: The P1003.7 group should be approached with the possibility of adding a virtual memory support interface. A sample interface could be drafted using HP-UX chatr() command as an example. If the first approach fails due to lack of support, then the interface should be added to the military standard. Consideration should be given to making the interface optional based on arguments outlined under alternative 1 above.

3.12.2 Virtual Space Locking

The requirement for Virtual Space Locking (12.2) is directly met by the 1003.4 Memory Locking functions.

3.12.3 Dynamic Memory Allocation and Deallocation

This unfulfilled requirement is classified as "a" (essential).

Requirement: The OSIF shall provide for allocation of a block of virtual or physical memory of the size specified and for deallocation of a previously allocated block. Note that this requirement has particular relevance for Ada applications, as specified in paragraph 3.16.15. Changes to the recommendations should take that fact into account.

Description of Delta: Memory management was purposely omitted as a separate 1003.1 function. Instead, it is included in Section 8 of 1003.1-1990 as part of the standard by virtue of being embodied in C language functions such as calloc, malloc, realloc, and free. Thus, 1003.1 relies on the language to provide memory management. However, it has become standard practice, considering the fact that most systems support shared memory and memory of several different types, to utilize the 1003.4 facilities of Memory Mapped Files to support memory allocation. Specifically, the 1003.4 mmap() function may be applied to a descriptor obtained by opening a special name associated with an allocator for a given type of memory; such a call then allocates the requested amount of that type of memory and returns a handle to that memory. The munmap() function provides the ability to deallocate memory allocated in this way. P1003.4d will specify these additional semantics plus additional interfaces necessary for use of 1003.4 mmap() and munmap() in this way, including the ability to allocate and share typed memory.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. A chapter for P1003.4d which will provide this capability has been drafted. Therefore, the obvious approach is to closely monitor this chapter to ensure that it continues to support the allocation OSSWG requires; when the chapter becomes stable, it will be entered into the draft, and the requirement will then be met by P1003.4d.

Recommendation: OSSWG should continue to support the efforts to complete the Typed Memory interfaces in P1003.4d.

3.12.4 Dynamically Protecting Memory

This unfulfilled requirement is classified as "a" (essential).

Requirement: The OSIF shall provide the ability to query and set memory-protection attributes.

Description of Delta: The POSIX standard provides dynamic memory protection for shared memory through the open() interface in 1003.4. The protection can be changed at runtime by closing and reopening the shared memory connection. There is no provision for protection of arbitrary blocks of memory or when allocating dynamic memory. P1003.4d contains a draft chapter for Typed Memory allocation implemented via mmap(). Since there is already control of mapped memory protection for all objects mapped via mmap(), this requirement will be directly met once the Typed Memory chapter of P1003.4d is approved by the working group.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. POSIX only provides for the protection of mapped or shared memory. There are no POSIX interface to query or set the memory protection attributes of other types of memory. The P1003.1a standards group has been discussing this issue. The 1003.4 group has drafted a chapter for P1003.4d which provides for Typed Memory allocation and associated protection via mmap() and mprotect().

2. Develop a new military standard. It is suggested that the interface be modeled after the Memory Locking interface in 1003.4 or on de facto UNIX or industry standard(s), if any exist.

Recommendation: Continue to support adoption of the P1003.4d Typed Memory allocation capabilities.

3.12.5 Shared Memory

This unfulfilled requirement is classified as "b" (highly desirable).

Requirement: The OSIF shall support concurrent access, by several processes, to specified areas of physical memory, whether or not the involved processes exist on a single processor or multiple processors.

Description of Delta: POSIX 1003.4 provides a set of interfaces for creating, attaching to, and deleting shared data regions. The requirement, however, specifies that both the data and the code regions need to be shared. The ability to share code is useful for libraries and certain utilities and could be a pre-runtime interface. Even though it is not explicitly stated that multi-processor shared memory is supported, there is nothing in the standard that precludes it.

POSIX also provides several interface alternatives for resolving contention during access to shared memory. These include Counting Semaphores in 1003.4 and P1003.20, and Mutexes and Condition Variables in P1003.4a. Mutexes and Condition Variables were designed particularly for processes that share memory. The POSIX Standardized Profile for Multiprocessing Systems (1003.14) has also proposed Reader/Writer Lock and Spin Lock interfaces. Since 1003.14 is a profile, it can not specify interfaces that are not defined in other standards. However, 1003.4d is considering inclusion of these interfaces in its specification.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. The POSIX shared data interfaces are found in the 1003.4 standard. There is no interface to specify shared code. The HP-UX operating system provides an interface to specify code as sharable. It is the same chatr() command referenced in 3.12.1. The most logical place for this type of interface seems to be P1003.7.

2. Develop a new military standard. It is suggested that any new military standard be based on de facto UNIX or industry standard(s), if any exist.

Recommendation: Recommend that this requirement be linked with requirement 12.1 and presented to the P1003.7 standards group. The HP-UX chatr() interface could be used as an example.

3.12.6 Allocate, Deallocate, Mount, and Dismount Services

This unfulfilled requirement is classified as "a" (essential).

The requirement for Allocate, Deallocate, Mount, and Dismount Services (12.6) is partially covered by the 1003.1 Control Operations on Files (file descriptors).

Requirement: The OSIF shall support the allocation of devices to processes and subsequent deallocation of these devices. For devices with removable media, the OSIF shall also support mounting and dismounting of media.

Description of Delta: 1003.1 provides allocate and deallocate functionality through the fcntl() interface. POSIX does not yet provide mount/dismount functionality. Refer to 3.7.10 for further discussion of this delta.

Resolution Alternatives: Same as requirement 7.10.

Recommendation: Same as requirement 7.10.

3.12.7 Designate Control

This unfulfilled requirement is classified as "b" (highly desirable).

Requirement: The OSIF shall provide the means to designate responsibility for maintaining the status and determining the configuration of a system resource. This requirement was reevaluated by a small group, which decided that it was "b" (highly desirable).

Description of Delta: There is no provision in POSIX for designating control of system resources.

Resolution Alternatives:

1. Change wording of the OCD to read "shall support" instead of "shall provide." Requirement can then be satisfied by the fork(), exec(), and kill() interfaces in the 1003.1 standard.

2. Enhance existing POSIX interfaces to include this capability. This requirement is similar to 7.1 device-driver availability . OSSWG recommended that these requirements be pursued in the 1003.4 or 1003.7 standards groups. Could combine 12.7 with solution to 7.1. Any solution needs to be compatible with solution to 12.8 release control.

Recommendation: Recommend this requirement be pursued with 7.1, and 12.8 OSSWG requirements in 1003.4 or 1003.7 standards groups.

3.12.8 Release Control

This unfulfilled requirement is classified as "b" (highly desirable).

Requirement: The OSIF shall provide the means to release a previously assumed system resource status and configuration responsibility. This requirement was reevaluated by a small group, which decided that it was "b" (highly desirable).

Description of Delta: See 3.12.7.

Resolution Alternatives: See 3.12.7.

Recommendation: See 3.12.7.

3.12.9 Allocate Resource

This unfulfilled requirement is classified as "a" (essential).

Requirement: The OSIF shall provide a means to designate particular process resources for use by a particular process.

Description of Delta: There is no provision in POSIX for allocating resources. Examples of units of system resources are I/O channel, a block of physical memory, response to specific class of hardware interrupt, a breakpoint register, a co-processor user identifier, and a connection over a LAN.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. Typically, UNIX resources such as files, devices, and network connections have been referred to under the general

description of a file. A logical to physical connection is created and referenced by a file descriptor. The same concept could be extended to include a number of different resources, particularly the ones of interest to OSSWG. The new interface(s) could be added to P1003.1a or 1003.4.

2. Develop a new military standard. This is a less acceptable alternative than 1 because it is external to the OSIF baseline. It is suggested that any new military standard be based on de facto UNIX or industry standard(s), if any exist.

Recommendation: Recommend that P1003.1a and 1003.4 be approached about extending definition of file to include all resources needed by OSSWG and provide interfaces to open, close, and lock these resources. OSSWG needs to be more specific on the scope of this requirement. The same resolution should be applied to requirement 12.10.

3.12.10 Deallocate Resource

This unfulfilled requirement is classified as "a" (essential).

Requirement: The OSIF shall provide a means to relinquish particular process resources from a particular process.

Description of Delta: See 3.12.9.

Resolution Alternatives: See 3.12.9.

Recommendation: See 3.12.9.

3.12.11 System Resource Requirements Specification

This unfulfilled requirement is classified as "b" (highly desirable).

Requirement: The OSIF shall provide the ability to specify system resource requirements. This requirement was reevaluated by a small group, which decided that it was "b" (highly desirable).

Description of Delta: There is no provision in POSIX for specifying system resource requirements.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. The concept of system resource requirements specification is not presently in any of the POSIX standards. The P1003.7 group would probably be the most receptive to the addition of an interface of this type.

2. Develop a new military standard. This is a less acceptable alternative than 1 because it is external to the OSIF baseline. It is suggested that any new military standard be based on de facto UNIX or industry standard(s), if any exist.

3. Submit a new POSIX PAR (System Resource Management) to do this work.

Recommendation: The 1003.7 group should be approached with the possibility of adding a system resource requirements specification interface. A sample interface could be drafted from examples from other operating systems that provided this functionality in a more complete manner. A backup position, should 1003.7 be unable or unwilling to take on this interface, would be alternative 3, submission of a new PAR for System Resource Management.

3.12.12 System Resource Capacity

This unfulfilled requirement is classified as "b" (highly desirable).

Requirement: The OSIF shall provide a query of the storage or workload capacities of the system resources. This requirement was reevaluated by a small group, which decided that it was "b" (highly desirable).

Description of Delta: There is no provision in POSIX for specifying system resource capacity. P1003.4d Typed Memory interfaces, when drafted, may allow applications to query a typed memory pool for the maximum amount of memory which can be allocated; However, this is unique to typed memory pool resources, not a generalized capability.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. The system resource capacity requirement is provided by the 1003.2 standard in an incomplete way through commands such as du and df. OSSWG has chosen to avoid inclusion of 1003.2 in the OSIF. The P1003.7 group would probably be the most receptive to the addition of an interface of this type.

2. Develop a new military standard. This is a less acceptable alternative than 1 because it is external to the OSIF baseline. It is suggested that any new military standard be based on de facto UNIX or industry standard(s), if any exist.

3. Submit a new POSIX PAR (System Resource Management) to do this work.

Recommendation: The 1003.7 group should be approached with the possibility of adding a system resource capacity interface. A sample interface could be drafted using 1003.2 examples and examples from other operating systems that provided this functionality in a more complete manner. A backup position, should 1003.7 be unable or unwilling to take on this interface, would be alternative 3, submission of a new PAR for System Resource Management. OSSWG should continue to support the drafting, refinement, and balloting of the P1003.4d Typed Memory facilities.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
12.1	Partially	Insertion	a
12.2	Yes	None	-
12.3	Partially	Insertion	a
12.4	Partially	Insertion	a
12.5	Partially	Insertion	b
12.6	Partially	Insertion	a
12.7	No	Insertion	b
12.8	No	Insertion	b
12.9	No	Insertion	a
12.10	No	Insertion	a
12.11	No	Insertion	b
12.12	No	Insertion	b

3.13 SYNCHRONIZATION AND SCHEDULING INTERFACES

In general, the POSIX standards support service class 13 synchronous and scheduling interfaces in a substantially complete way.

3.13.1 Process Synchronization

The requirements for Process Synchronization (13.1) are directly met by 1003.4, P1003.4a and P1003.4b. Pthreads appears to fully satisfy this requirement by providing mutex and condition variable primitives for synchronization among threads within the same process. This includes semaphores, signals, events, message queues, etc., for synchronization among threads in different processes.

3.13.2 Mutual Exclusion

The requirements for Mutual Exclusion (13.2) are fully met by 1003.1, 1003.4, P1003.4a, and P1003.4b. Both mutexes and semaphores support mutual exclusion among cooperating processes and/or cooperating threads, and P1003.4b extends both of these such that the waits may time out. Lock files are supported by the 1003.1 open() interface.

3.13.3 Cumulative Execution Time of a Process

The requirements for Cumulative Execution Time of a Process (13.3) are directly met by 1003.1 Process Times and P1003.4b CPU Time Clocks.

3.13.4 Attach a Process to an Event

This requirement is directly met by 1003.1 Signals as extended by 1003.4 to Queued Signals and as further extended by P1003.4a to operate in a multi-threaded process; and by P1003.4b Interrupt Control interfaces.

3.13.5 Services Scheduling Information

This unfulfilled requirement is classified as "d" (re-evaluate).

The requirement for services scheduling information (13.5) is not supported by the POSIX standards at all.

Requirement: The OSIF shall support the ability for a process to specify its performance requirements for services.

Description of Delta: This requirement implies that, in order to guarantee timely completion of a complex service across a distributed system, the application requires an upper bound on time for that service. This is seen as similar to the "time-value" function associated with a service interface in operating systems such as Alpha. Such a function serves to define the urgency of a particular request separately from the CPU scheduling policy for the requesting process. Currently, OSSWG does not perceive this issue as being addressed by any POSIX working group.

Resolution Alternatives:

1. Enhance existing POSIX interfaces to include this capability. This may already be possible due to the open nature of the 1003.4 and 1003.4a process/thread scheduling interfaces; that is, if a new scheduling policy could be defined in which a process could maintain a

transaction scheduling attribute, and if this policy were included among the selectable policies, the requirement might be satisfied. Because such a policy may not be well understood by the industry, POSIX has decided to leave such a policy out of the standards for now, while leaving a method for its future insertion.

Also, 1003.11 needs to be further queried to determine if this capability conforms to its charter, since outside of 1003.11, most interfaces do not address the special needs of atomic transactions, especially over a distributed network. Therefore, it might be more appropriate that such transactions be addressed by 1003.11 rather than 1003.4. This is the most suitable alternative because the need for this has already been recognized by VITA and by several other vendors.

2. Assume a standard outside of POSIX. It is difficult to understand the scope of this requirement sufficiently to rule out various higher level distributed processing interfaces built on top of existing operating systems, such as ISIS. However, as stated, it seems to imply a bounded time that could be achieved only if the POSIX kernel were cooperating.

Recommendation: OSSWG recommends alternative 1. However the 1003.11 working group has been dissolved and cannot be used to resolve this delta. Furthermore, the 1003.4 working group has rejected this requirement for inclusion in P1003.4d because of immaturity of existing practice. OSSWG should pursue this requirement in the Realtime Distributed Systems Communication working group 1003.21 at such time in the future as existing practice can be identified. The 1003.21 working group is currently evaluating how such information might be applied to network service interfaces. OSSWG should re-evaluate this requirement based on the 1003.21 findings, both as applied to distributed systems, and if applicable, non-distributed systems.

3.13.6 Scheduling Delay

This requirement is functionally identical to requirement 9.7 and has no delta.

3.13.7 Periodic Scheduling

The requirement for Periodic Scheduling (13.7) is fully met by 1003.1 Signals, alarm(), and sleep(); 1003.4 Timers and High Resolution Sleep; P1003.4a Timed Condition Wait; and P1003.4b Sporadic Server and Interrupt Control. The POSIX approach of specifying performance metrics provides a mechanism for the jitter to be determined for a particular implementation. However, performance metrics are currently non-normative text in 1003.4 and P1003.4a; therefore OSSWG should support future POSIX projects which seek to standardize performance metrics.

3.13.8 Multiple Scheduling Policies

The requirement for Multiple Scheduling Policies (13.8) is covered fully by 1003.4, P1003.4a, and P1003.4b Execution Scheduling interfaces.

3.13.9 Selection of a Scheduling Policy

The requirement for Selection of a Scheduling Policy (13.9) is covered fully by 1003.4, P1003.4a , and P1003.4b Execution Scheduling interfaces.

3.13.10 Modification of Scheduling Parameters

The requirement for Modification of Scheduling Parameters (13.10) is covered fully by 1003.4, P1003.4a, and P1003.4b Execution Scheduling interfaces.

3.13.11 Precise Scheduling (Jitter Management)

The requirement for Precise Scheduling (13.11) is fully met by 1003.4, P1003.4a, and P1003.4b Execution Scheduling, Timers, and Interrupt Control interfaces. The POSIX approach of specifying performance metrics provides a mechanism for the latency to be defined for a particular implementation. However, performance metrics are currently non-normative text in 1003.4 and P1003.4a; therefore OSSWG should support future POSIX projects which seek to standardize performance metrics.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
13.1	Yes	None	-
13.2	Yes	None	-
13.3	Yes	None	-
13.4	Yes	None	-
13.5	No	Insertion	d
13.6	Yes	None	-
13.7	Yes	None	-
13.8	Yes	None	-
13.9	Yes	None	-
13.10	Yes	None	-
13.11	Yes	None	-

3.14 SYSTEM INITIALIZATION AND REINITIALIZATION INTERFACES

This service class is partially supported by 1003.1, 1003.4, P1003.7, and P1003.8.

All three requirements from this service class are classified as "a" (essential). POSIX generally supports these requirements only as they might apply to a shore-based information processing system with a system administrator in charge of overall system operation, and time-shared users in charge of initiating and terminating independent programs. This concept must be extended to support embedded real-time systems in which individual programs and overall system operation are controlled by software, hardware, or other nodes on a distributed processing network, rather than by a person. Performance also is an issue largely ignored by 1003.7; system reinitialization may imply an operation that must be completed in seconds or milliseconds, rather than minutes.

3.14.1 Image Load

This unfulfilled requirement is classified as "a" (essential).

The Image Load requirement (14.1) can be supported by 1003.1, Process Creation and Execute a File, but not in the traditional sense of program or boot load. P1003.7, when complete, would fully support this function in the Machine Class and System Class. File and Directory Services of 1003.1 might also be required.

Requirement: The OSIF shall provide the capability to perform initial and reinitial executable image load (including data) both locally and remotely to and for each and all processor(s) throughout a system.

Description of Delta: The POSIX standard is based on the traditional UNIX paradigm where all processes are ultimately children of the root process. The emerging computing environment is one of multiple quasi-independent processors on the same backplane, or network, which must communicate and interact through OS services. One of the extensions of this multi-processor environment is that the OS must be able to start and restart each of the computing resources available to it.

In the 1003.1 standard, the ability to spawn a child process and to start a new execution are described. These services will partially meet the requirements of Image Loading. The issues that are not addressed by these sections of 1003.1 are:

1. Loading and executing on a remote processor(s).
2. Loading and executing on another local processor(s).
3. Reloading the data area for each (re)initialization.

Recommendation: It is recommended that a new interface be created either by the 1003.1 or 1003.7 group. The interface would be very similar to the various exec() interfaces that exist in 1003.1. This would essentially be a remote execution command, sending a "new process image file," including both executable and data areas, to another processor to be executed.

Note: The 1003.7 standards need to be influenced beyond their current focus to become true resources manager standards, including management of both remote and local resources. This change would help meet the OCD requirements for not only section 20.14.1, but also 20.14.2 and 20.14.3 (and possibly many others).

3.14.2 System Initialization and Reinitialization

This unfulfilled requirement is classified as "a" (essential).

The System Initialization and Reinitialization requirement (14.2) can be supported by the entire sections on Process Primitives and Process Environment of 1003.1. 1003.1 File and Directory Services might also be required.

P1003.7 fully supports this function in the Interoperability Class, Machine Class, System Class, Network Class, Authentication Class, Authorization Class, Software Class, and Backup Class. P1003.7, when complete, could become the NGCR resources management standard as a function of system administration. With some influence and direction, it could be expanded, either as a profile or standard, to support the necessary NGCR resources management functions. Additional support will be provided by 1003.4, Clocks and Timers, and P1003.8, Process Creation.

Requirement: The OSIF shall support the capability to initialize and reinitialize all system resources.

Description of Delta: It is important to clarify that "system resources" as mentioned in the OCD are ALL computing resources including, but not limited to, printers, disk drives, external and shared memory, co-processors, tape drives, and display systems.

1003.1 allows for process creation and signal generation/reception. These two components could be made to help in performing system (re)initialization. The ability to start processes on remote processors (see discussion for OCD section 20.14.1) could cover the need to (re)initialize some resources. Other resources may be able to receive POSIX signals that would cause (re)initialization.

1003.1 allows for collecting system information and parameters. This would allow the OS to gain information about system resources so that it would know when and what needed to be (re)initialized.

P1003.7 seems to have the outline to become the NGCR resource management standard, but it needs to be further developed.

Recommendation: OSSWG needs to influence the POSIX standards groups (both 1003.1 and 1003.7) to create the ability for the operating system to (re)initialize the system resources. This capability really doesn't exist in the POSIX standards but is an absolute requirement for OSSWG.

3.14.3 Shutdown

This unfulfilled requirement is classified as "a" (essential).

The Shutdown requirement (14.3) can be supported by 1003.1, Wait for Process Termination and Terminate a Process.

Requirement: The OSIF shall provide the capability to perform planned, orderly shutdown at the local and remote levels for each and all processor(s) throughout a system.

Description of Delta: 1003.1 outlines how POSIX processes can stop, but offers no capability for forcing the termination of one process from another non-related process.

Recommendation: OSSWG should influence the POSIX standards to include the capability to force a process termination on remote processors. This change can either be implemented in 1003.1, or added to P1003.7 as part of the resources management standard.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
14.1	Partially	Insertion	a
14.2	Partially	Modification	a
14.3	Partially	Insertion	a

3.15 TIME SERVICES INTERFACES

In general, the POSIX standards substantially support the time services.

The time services requirements selection of a primary reference clock (15.4), and location of the primary reference clock (15.5) are not specifically supported in POSIX. In the event of the loss of the primary reference clock the OSIF does not provide a means to locate a new primary reference clock when needed.

The Ada language calendar package, Calendar, and the 1003.5 Ada package, POSIX_Calendar, are equivalent in their functionality. They have the same provisions for getting the time and performing operations against that time. The 1003.5 package POSIX_Calendar has one advantage in that it has a procedure to override the system's default time zone through the TZ environment variable.

3.15.1 Read Selected Clock

The requirement for Read Selected Clock (15.1) for timer services, and for precision is directly and completely met by 1003.4 Clocks and Timers. In addition, there are interfaces in 1003.1, 1003.2, P1003.4b, and potentially P1003.7 that partially meet the requirements to read a clock.

System Time (paragraph 4.5.1, 1003.1 and paragraph 4.4.1, 1003.5) provides access to a time-of-day clock, with precision to a hundredth of a second. Process Times functions (paragraph 4.5.2, 1003.1 and paragraph 4.2, 1003.5) return the number of clock ticks since the beginning of a particular process. The Clocks and Timers interface described in 1003.4 and P1003.20 allows multiple clocks to be defined. Every system that supports this interface must define at least the system real-time clock. The interface provides for potential resolution down to a nanosecond.

3.15.2 Set Selected Clock

The requirement for Set Selected Clock (15.2) for timer services, and for precision is directly and completely met by 1003.4 Clocks and Timers and P1003.4b CPU Time Clocks and Device Control. In addition, P1003.7 can address setting a clock.

System Time (paragraph 4.5.1, 1003.1 and paragraph 4.4.1, 1003.5) does not allow for setting the time-of-day clock. All clocks defined by the Clocks and Timers interface in 1003.4 and P1003.20 may be set as well as read.

3.15.3 Synchronization of Selected Clocks

The requirements for Synchronizing Selected Clocks (15.3) for timer services is directly and completely met by 1003.4 Clocks and Timers and P1003.4b Device Control.

Synchronization of selected clocks is supported, through the combination of the get and set functions and the identification of the clocks throughout the system. The Device Control interface in P1003.4b allows getting and setting clocks located on an external device.

3.15.4 Select a Primary Reference Clock

This unfulfilled requirement is classified as "a" (essential).

The Selection of a Primary Reference Clock is not specifically supported in POSIX since the specific wording of our requirements implies the ability to dynamically reconfigure the system wide clock and define another system wide clock.

The requirement for Selection of a Primary Reference Clock (15.4) is only partially met by 1003.4 and P1003.4b Clocks and Timers. Selection of a primary can only be done by virtue of an application's use of a specific clock reference which must be initially defined potentially by 1003.7.

There is no means to set or change the default in a dynamic way.

<u>Requirement:</u> The OSIF shall support the ability to select a primary reference clock for the system.

<u>Description of Delta:</u> POSIX working group 1003.21 has identified a requirement for access to global time in their requirements document. They have requested a new PAR on time management to be assigned to the 1003.21 working group. Pending approval of this PAR and initiation of a draft standard on time management, POSIX does not yet address this issue.

<u>Recommendation:</u> The OSSWG should support the 1003.21 working group's time management proposals through standardization to ensure that this requirement is met.

3.15.5 Locate the Primary Reference Clock

This unfulfilled requirement is classified as "d" (re-evaluate).

The Location of the Primary Reference Clock is not specifically supported in POSIX since the specific wording of our requirements implies the ability to dynamically reconfigure the system wide clock and define another system wide clock.

The requirement for Location of the Primary Reference Clock (15.5) is limited to the predefined system wide clock. The location of another primary reference clock in the event of a failure of the predefined system wide clock is not covered in any of the POSIX documents. This failing, as well as the partial coverage addressed in the previous paragraph, is attributable to the lack of real attention to the needs of distributed systems and the demands they place on time services.

Requirement: The OSIF shall support the ability to locate the primary reference clock for a system.

Description of Delta: The Distributed Time Services requirements in the 1003.21 working group requirements document refer to access to a distributed system clock without reference to its location. This 1003.21 working group requirement should preclude the need for OSSWG requirement 15.5.

Recommendation: The OSSWG should support the 1003.21 working group through standardization of it's proposed draft standard. OSSWG should consider changing this requirement to be more in line with the 1003.21 requirement.

3.15.6 Timer Services

The Timer Services requirement (15.6) is fulfilled by the POSIX standards 1003.1, 1003.4, P1003.4a, and P1003.4b. The Alarm, Timer, and Interrupt Control interfaces in these standards, plus the related capabilities to await signals and interrupts satisfy this requirement.

3.15.7 Precision Clock

Precision Clock (15.7) is fully supported by the 1003.4 timespec structure for Clocks and Timers, which permits resolutions down to 1 nanosecond.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
15.1	Yes	None	-
15.2	Yes	None	-
15.3	Yes	None	-
15.4	Partially	Modification	a
15.5	No	Insertion	d
15.6	Yes	None	-
15.7	Yes	None	-

3.16 ADA LANGUAGE SUPPORT

The POSIX interface reflects fundamental aspects of UNIX and, in turn, the support it offers to Ada implementations must be seen in that light. UNIX was designed and built to support a multiple-user interactive environment. Its whole notion and implementation of process reflects the need to supply resources to users equitably, while protecting them from accidental interference with one another. In particular, processes are the only objects where concurrency is applicable, and they comprise single threads of control within unique address spaces. Further, fundamental aspects of the design of the system reflect the assumption that text processing and I/O would be important aspects of the processes supported, and that the processes would be running on single-processor computers. (The more general applicability of many recent implementations has had to deal with this orientation of UNIX.)

The consequence of these design elements of UNIX and POSIX is that the general POSIX definition, 1003.1, does not offer much positive support for the implementation of Ada systems. In practice, an Ada runtime on POSIX, as on UNIX, will not be able to use its fundamental services (such as process management, synchronization, and scheduling) to provide Ada semantics directly.

The fundamental reason for this lack of support is that POSIX processes are unsuitable as a mapping for Ada tasks. Processes do not share memory, and tasks do. Processes can continue executing even when their parents have terminated, while this is not possible for Ada tasks. Processes inherit their parents' attributes in ways that Ada tasks do not. Switching contexts between processes has more overhead than would be desirable for tasks.

This does not mean that Ada cannot be implemented in a POSIX system. It simply means that the Ada runtime will need to do most of its own work to implement Ada semantics. Also, there are some instances in which POSIX, like UNIX, will get in the way; such as the fact that making a request for I/O blocks an entire process (read Ada program). This is understandable in a multi-user interactive environment, but is unsuitable in many Ada applications.

The real-time extensions (1003.4), however, and particularly the threads extensions (P1003.4a), are more helpful. First of all, synchronization primitives (semaphores, mutexes, and condition variables) are made available. Second, threads appear to provide a suitable mapping to Ada tasks, such that it would be feasible to assume that a POSIX implementation which included the real-time and threads options could provide task management and scheduling for an Ada runtime environment. Other services could be used directly to implement Ada semantics as well.

In general, in some instances, Ada semantics will be implementable by inserting calls to POSIX real-time and thread services directly into the compiled code. On the other hand, in most instances, the Ada runtime library will need to carry out extra-POSIX activities; sometimes with the assistance of calls to POSIX services, and occasionally completely on its own. The threads extensions (P1003.4a) document outlines how an Ada system might map tasks on the threads primitives.

In this section it is assumed that the Ada binding to POSIX (1003.5) is a reflection of 1003.1, rather than the provision of additional support for Ada. 1003.5 provides for Ada I/O support in addition to the POSIX I/O and adds services to relate the two types of I/O.

In general, the POSIX standards support service class 16, Ada language support interface, in a substantially complete way for the POSIX (P1003.4a) thread model and in a rudimentary way for the POSIX process model (1003.1).

The requirements for the Ada task model are met in a fairly direct way by the POSIX thread model. The support of tasks in isolation (i.e., create (16.1), terminate (16.5), etc.) is quite direct. The support of Ada rendezvous and selective waiting is complete, but it requires extensive, specialized composition of POSIX services.

A number of the OSSWG requirements for the support of Ada are requirements for services to be provided by the run-time system. These requirements include access to task characteristics (16.9), access to a precise real-time clock (16.11), access to the time-of-day clock (16.12), dynamic task priorities

(16.13), memory management (16.15), and exception raising (16.19). The POSIX thread model supports these run-time system requirements with a few exceptions.

The unfulfilled requirements in this section are duplications of requirements in previous sections. They are requirements that have special relevance for Ada language applications, but if they are fulfilled by the OSIF in general, they will be fulfilled also for Ada applications. It does not seem wise to duplicate the exposition of the issues, since it would incur the dangers of duplicate maintenance. These sections will therefore refer to the sections that define the issues and recommend actions.

Some general recommendations are appropriate, however, to ensure that the solutions derived for the deltas are appropriate for Ada applications:

1. The OSSWG should remain active in the 1003.5 (Ada Bindings) group to ensure that the Ada bindings to POSIX interfaces are adequate to fulfill the requirements of NGCR Ada applications.

2. The discussions of the specified deltas in previous sections should also make reference to the Ada-specific section to ensure that the delta is resolved. Even in the unlikely event that it were to be decided that there is no general need for the functions, there is still a requirement in an Ada context. This judgment should not be lost.

3. The OSSWG should follow the progress of Ada9X, since there is some indication that language changes will be made that will have impact on requirements defined in this section.

3.16.1 Create Task (Ada)

The requirement for Create Task (16.1) is met by P1003.4a. Refer to the Pthreads discussion in 3.9.1.

3.16.2 Abort Task (Ada)

This unfulfilled requirement is classified as "a" (essential).

This requirement is unfulfilled for the same reason that requirement 9.2 is unfulfilled; that is, there is no interface provided in P1003.4a to unconditionally terminate a thread. Refer to section 3.9.2 for recommendations.

3.16.3 Suspend Task (Ada)

The requirement for Suspend Task (16.3) is met by 1003.4 and P1003.4a. Refer to the Pthreads discussion in 3.9.5.

3.16.4 Resume Task (Ada)

The requirement Resume Task (16.4) is met by 1003.4 and P1003.4a. Refer to the Pthreads discussion in 3.9.6.

3.16.5 Terminate Task (Ada)

The requirement Terminate Task (16.5) is addressed by P1003.4a Thread Cancellation. Ada task termination semantics imply cooperation form the terminating task; thus thread cancellation provides a suitable interface to meet this requirement in spite of its inability to unconditionally terminate an uncooperative task.

3.16.6 Restart Task (Ada)

This unfulfilled requirement is classified as "d" (re-evaluate).

The proposed Ada extension to support "Restart Task" (16.6) is not supported by either the POSIX process or thread model. This requirement is perhaps the most controversial of the proposed Ada extensions.

Restart Task (Ada) (16.6) is required for OSSWG if seen independently from its connection to support for Ada; as such it is dealt with in requirement 9.13 (save/restart process). On the other hand, the requirement does not relate to the current definition of the Ada language and therefore should be reevaluated as to whether it should be duplicated in this section. Some people in the Ada community have suggested that the language should be modified to allow more direct access to these functions, and it is possible these functions will be included in the next revision, now called Ada-9X. Thus, this requirement is classified as "d" (re-evaluate).

Requirement: The OSIF shall support the capability to restart the execution of an Ada task at a point immediately following its elaboration code.

Description of Delta: This requirement reflects a need to provide extensions to the current Ada language standard. OSSWG should give careful study to the appropriateness of the requirement, monitor the progress of language modification efforts, and propose further additions to the POSIX standard, either as changes to P1003.4a or inclusion in P1003.4b.

Recommendation: See section 3.9.13. OSSWG should re-evaluate this requirement this requirement based on Ada-9X capabilities.

3.16.7 Task Entry Calls (Ada)

Some of the claims found in P1003.4a regarding support of Task Entry Calls (16.7) cannot be fully accepted without further proof through implementation and validation. The 1003.5 working group has submitted objections to .4a which, if resolved, will allow Ada tasks to be mapped to .4a threads. If not resolved, and without mapping Ada tasks to threads, Task Entry Calls can still be achieved via other POSIX interfaces, but with reduced performance.

3.16.8 Task Call Accepting/Selecting

Some of the claims found in P1003.4a regarding support of accepts (16.8) cannot be fully accepted without further proof through implementation and validation. The 1003.5 working group has submitted objections to .4a which, if resolved, will allow Ada tasks to be mapped to .4a threads. If not resolved, and without mapping Ada tasks to threads, Task Call Accepting/Selecting can still be achieved via other POSIX interfaces, but with reduced performance.

3.16.9 Access Task Characteristics (Ada)

The requirement to Access Task Characteristics (16.9) is supported by Clock and Timer Functions of 1003.4, Thread Management and Thread Cancellation of P1003.4a, and also Thread Scheduling Functions and CPU-Time Clock of P1003.4b.

3.16.10 Monitor Task's Execution Status (Ada)

This unfulfilled requirement is classified as "a" (essential).

Monitor Task's Execution Status (Ada) (16.10) is required by OSSWG and is dealt with independently in requirements 9.11 (Examine Process Status) and 13.3 (Cumulative Execution Time of a Process).

This requirement is important to the spirit of the Ada standard and to real-time applications. OSSWG should propose further additions to the POSIX standard, either as changes to P1003.4a or inclusion in P1003.7.

Requirement: The OSIF shall support the ability to monitor a task's execution status, in particular, the amount of accumulated CPU time that has been used by the task.

Description of Delta: The requirement for Monitor Task's Execution Status (16.10) is not met by 1003.1, 1003.4, or P1003.4a. Since Ada tasks must be mapped onto POSIX threads the standard process primitives are not available to support this requirement. 1003.2 has not been extended to address thread status. P1003.4b does allow access to the CPU time used by a thread.

Recommendation: See section 3.9.11.

3.16.11 Access to a Precise Real-Time Clock (Ada)

The requirement to Access a Precise Real-Time Clock (16.11) is covered in sections 3.15.1, 3.15.2, and 3.15.7. There is no additional requirement peculiar to Ada.

3.16.12 Access to a TOD Clock (Ada)

The requirement to Access a Time of Day Clock (16.12) is covered in sections 3.15.1, 3.15.2, and 3.15.7. There is no additional requirement peculiar to Ada.

3.16.13 Dynamic Task Priorities (Ada)

The Dynamic Task Priorities requirement (16.13) is provided by both P1003.4a and P1003.4b, with interfaces to get and set thread scheduling parameters.

3.16.14 Scheduling Policy Selection (Ada)

Scheduling Policy Selection (16.14) is also required by OSSWG and is dealt with independently in requirement 13.9 (Selection of a Scheduling Policy). While not directly visible to Ada applications, this interface may be critical to the implementation of an Ada run-time.

This requirement reflects a need to provide extensions to the current Ada language standard. OSSWG should give careful study to the appropriateness of the requirement and monitor the progress of language modification efforts.

The Scheduling Policy Selection (16.14) requirement is fully supported by 1003.4 and P1003.4a (1003.1 provides no support for scheduling policy selection). Reference P1003.4a, "Thread Creation Scheduling Attributes," "Thread Scheduling;" 1003.4, "Execution Scheduling;", and P1003.4b, "Process and Thread Scheduling Functions."

A number of the OSSWG requirements for Ada language support are actually requirements for Ada extensions that may or may not become a part of the language standard in the future. In the case of scheduling policy selection (16.14), the 1003.4, P1003.4a , and P1003.4b interfaces provide extensive support.

Requirement: The OSIF shall support the capability to get and set the policy that is to be used to schedule Ada tasks.

Recommendation: There is no longer an OSSWG delta per-se, but rather only an Ada delta. It is recommended that OSSWG address this issue as a whole.

3.16.15 Memory Allocation and Deallocation (Ada)

This unfulfilled requirement is classified as "a" (essential).

Memory Allocation and Deallocation (Ada) (16.15) is required by OSSWG and is dealt with independently in requirement 12.3 (Dynamic Memory Allocation and Deallocation). It is particularly important to the implementation of Ada systems.

This requirement is unfulfilled for the same reason that requirement 12.3 is unfulfilled; that is, there are no sufficiently flexible interfaces provided in POSIX for dynamic memory allocation and deallocation, but the Typed Memory interfaces in P1003.4d will satisfy this requirement once a draft is produced. There is no additional requirement peculiar to Ada. Refer to section 3.12.3 for recommendations.

3.16.16 Interrupt Binding (Ada)

This requirement is directly met by P1003.4b Interrupt Control.

3.16.17 Enable/Disable Interrupts (Ada)

Enable/Disable Interrupts (Ada) (16.17) is required for OSSWG if seen independently from its connection to support for Ada; as such it is dealt with in requirement 5.5 (Enable/Disable Interrupts). There is no longer a delta for requirement 5.5 because P1003.4b includes interfaces which provide mutual exclusion between an application and its interrupt handler. On the other hand, the requirement does not relate to the current definition of the Ada language and therefore should be reevaluated as to whether it should be duplicated in this section. Some people in the Ada community have suggested that the language should be modified to allow more direct access to these functions, and it is possible these functions will be included in the next revision, now called Ada-9X. Thus, this requirement is classified as "d" (reevaluate).

This requirement reflects a need to provide extensions to the current Ada language standard. OSSWG should give careful study to the appropriateness of the requirement and monitor the progress of language modification efforts.

A number of the OSSWG requirements for Ada language support are actually requirements for Ada extensions that may or may not become a part of the language standard in the future. In the support of Enable/Disable Interrupts (16.17), as described in the OSSWG requirements, a marginally satisfactory masking capability is provided in 1003.1, 1003.4, and P1003.4a as related to signals; but P1003.4b Interrupt Control provides a much more generic capability.

Requirement: The OSIF shall support the capability to enable and disable interrupts.

Recommendation: OSSWG should re-evaluate this requirement based on Ada-9X capabilities. There is no OSSWG delta per-se, but rather only an Ada delta.

3.16.18 Mask/Unmask Interrupts (Ada)

This unfulfilled requirement is classified as "c" (may be deferred).

Mask/Unmask interrupts (Ada) (16.18) is required for OSSWG if seen independently from its connection to support for Ada; as such it is dealt with in requirement 5.6 (Mask/Unmask Interrupts). On the other hand, the requirement does not relate to the current definition of the Ada language and therefore should be reevaluated as to whether it should be duplicated in this section. Some people in the Ada community have suggested that the language should be modified to allow more direct access to these functions, and it is possible these functions will be included in the next revision, now called Ada-9X. Thus this requirement is classified as "d" (reevaluate).

This requirement reflects a need to provide extensions to the current Ada language standard. OSSWG should give careful study to the appropriateness of the requirement and monitor the progress of language modification efforts.

A number of the OSSWG requirements for Ada language support are actually requirements for Ada extensions that may or may not become a part of the language standard in the future. In the support of Mask/Unmask Interrupts (16.18), as described in the OSSWG requirements, only a marginally satisfactory masking capability is provided in 1003.1, 1003.4, and P1003.4a as related to signals. The P1003.4b Device Control interface may be interpreted as a standard way to request a device to mask or unmask its interrupts.

Requirement: The OSIF shall support the capability to mask and unmask device interrupts.

Recommendation: Same as in section 3.5.6. There is no additional requirement peculiar to Ada.

3.16.19 Raise Exception (Ada)

Support for the Raise Exceptions requirement (16.19) is believed to be provided by a combination of services for signals within 1003.1, 1003.4 and P1003.4a, but this support has not yet been proven.

3.16.20 I/O Support (Ada)

The requirement for Ada Input/Output Support (16.20) is partially covered by 1003.1, 1003.4, P1003.4a, 1003.5, and P1003.20. 1003.1, 1003.4, and P1003.4a define the POSIX file support and I/O primitives. 1003.5 and P1003.20 provide the Ada binding to those POSIX features, as well as services to convert between the two versions. Support for Ada Low_Level_IO is provided by the P1003.4b Device Control interface.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	Unfulfilled Requirements Rating
16.1	Yes	None (1,3)*	-
16.2	Partially	Insertion (1,3)	a
16.3	Yes	None (1,3)	-
16.4	Yes	None (1,3)	-
16.5	Yes	None (1,3)	-
16.6	No	Insertion (2)	d
16.7	Yes	None (1,3)	-
16.8	Yes	None (1,3)	-
16.9	Yes	None (1,3)	-
16.10	No	Insertion	a
16.11	Yes	None	-
16.12	Yes	None	-
16.13	Yes	None	-
16.14	Yes	None (2)	-
16.15	No	Modification	a
16.16	Yes	None	-
16.17	Yes	None(2)	-
16.18	No	Modification (2)	c
16.19	Yes	None (1,3)	-
16.20	Yes	None	-

*1 Requires a solid commitment to 1003.4 and P1003.4a by the POSIX standards effort.

2 Requires coordination between the Ada language standard and the POSIX standard.

3 Awaiting proof of adequacy of POSIX interfaces.

4. BIG 6 DISCUSSION

This section analyzes of the extent to which the POSIX standards meet what the NGCR OSSWG has termed the "Big Six." This refers to six technology areas that the Navy's NGCR Program Office has stated as being of prime importance to future Navy systems. These areas as related to computer systems are Distribution, Real-Time, Fault-Tolerance, Security, Heterogeneity, and Ada.

4.1 DISTRIBUTED SYSTEMS

It was always a primary goal of NGCR in general, and the NGCR OS in particular, to support the wide variety of distributed architectures found in Navy systems. Such systems include anywhere from two to hundreds of homogeneous and/or heterogeneous processing and I/O nodes communicating either point-to-point or via a multi-level bus or network interconnection. Ideally, the operating system interface should provide distributed services in a portable manner, masking the actual method of interconnection and its associated protocols.

Operating system services related to distributed processing can be broadly classified as either explicit or implicit distribution. Explicit distribution implies that the application directs a request to a specific logically identified node; an example of explicit distribution is sending a message to a specified node or I/O subsystem and awaiting a reply. Implicit distribution, conversely, implies that the application is unaware of where in the distributed system a requested service is provided; examples of implicit distribution include file servers, name servers, and the like.

4.1.1 Distribution in UNIX

Traditionally, UNIX systems have been primarily implemented on single node, uniprocessor systems. When the need for operation in a networked environment became obvious (stimulated by the ARPANET research in the late 1970s and early 1980s), explicit distributed services first began to appear as shell and utility add-ons to the basic UNIX systems; such facilities as electronic mail and file transfer services were built on OS and vendor-specific implementations of Defense Advanced Research Projects Administration's (DARPA) TCP/IP networking protocol. Researchers at the University of California Berkeley developed a portable API suitable for interprocess communication within a single node or across nodes via networking protocols; this interface, called Sockets, became a de-facto standard API for networking applications, thereby allowing portable versions of these explicit services to be built as utility applications. AT&T developed a similar interface, XTI, for its System V variant of UNIX.

In recent years, additional utility level explicit distributed services have become standard in most UNIX systems. These include remote shell, remote login, remote talk, and finger services, all implemented using a client-server model at the UNIX application level, and utilizing the Sockets or XTI API to send and receive service-specific messages via service-specific sockets across distributed nodes. Even more recently, implicit distributed services have been integrated into some UNIX systems, such as network file system and domain name server capabilities. These achieve a level of application transparency by embedding the remote node identification in configurable operating system tables that are maintained by a system administrator but are otherwise of no concern to portable applications.

Very recent developments in transparent distributed database and information retrieval include the WAIS (Wide Area Information Server) and Internet Gopher systems which both provide a seamless local user interface to widely distributed information.

4.1.2 Distribution in POSIX

The POSIX working groups seek to standardize current practice in the UNIX community. The current working groups therefore focus on a protocol independent interface (P1003.12), transparent file access (P1003.8), directory services (1224.2), object management (1224), X.400 message handling (1224.1), and common OSI API & FTAM API (P1238) for distributed systems.

The protocol independent interface is currently based on the Berkeley Sockets and XTI de-facto standards. A new PAR (Real Time Distributed Systems Communication - 1003.21) has proposed extending these capabilities for realtime systems. Likewise, the other APIs are based on de-facto industry standards. While 1224 and P1238 are not strictly part of POSIX (1003), they are part of the IEEE PASC (Portable Applications Standards Committee), and meet, distribute documents, and generally coordinate with POSIX.

4.1.3 Distribution in NGCR OS

All NGCR OS distribution requirements are not called out explicitly as OSSWG requirements. While the network and communications interfaces service class specifies the lowest level requirements for internode communication over LAN, bus, and point-to-point hardware interconnects, distribution is implicitly required by a number of APIs in other service classes. Each and every OSSWG requirement must be interpreted in the following manner: If this requirement makes sense in a distributed context, then the NGCR OS must support it in that distributed context.

For example, Navy embedded systems traditionally support some form of interprocess communication among processes at separate nodes; thus, OSSWG requirement 9.8 (interprocess communication) requires distribution support. In this case, the OSSWG requirement is general enough to cover both explicit distribution (i.e., the application sets up the logical pathway between the processes) and implicit distribution (i.e., the application interface is no different whether the communication is internode or intranode).

As a counterexample, the OSSWG requirement for mutual exclusion (13.2) is typically not implemented across nodes in Navy embedded systems, at least not at the operating system level. The reason for this is that mutual exclusion primitives are intended to be a high performance, low contention method for guarding shared resources against inappropriate simultaneous access; this model becomes virtually useless over high latency internode communication paths. Resources sharable across multiple loosely coupled nodes occur quite infrequently and are typically guarded with other mechanisms such as monitors (server processes).

4.1.4 NGCR/POSIX Distribution Delta

During the OSSWG evaluations that led to the selection of POSIX as the baseline for the NGCR MIL-STD OSIF, evaluators were constantly aware that each OSSWG requirement might have different implications in a loosely or tightly coupled distributed system than in a simple uniprocessor system. Although there is no OSSWG service class dealing specifically with distribution, service classes 2, 4, 8, 12, and 14 contain requirements that deal specifically with the explicit nature of distributed systems. Most other service classes contain one or more requirements for which some NGCR distributed systems will undoubtedly need transparent (implicit) distribution. The OSSWG has not reached a consensus on exactly which POSIX interfaces should be transparently distributed. However, since POSIX is currently providing very little transparent distribution of services, the delta is likely to widen when such transparent service interfaces are identified.

4.2 REAL-TIME SYSTEMS

The primary application of the NGCR OS is in support of Navy air, surface, subsurface, and shore-based mission computer systems. The secondary application is in all other Navy computer systems, including software development, laboratory, and non-military functions. Virtually all of the primary applications and some of the secondary applications have real-time constraints ranging from "soft" to "hard" real time. UNIX operating systems have traditionally offered very poor support for users with real-time requirements. Faced with this dismal reputation, various UNIX vendors have offered a variety of nonstandard, nonportable real-time extensions to the UNIX kernel.

4.2.1 Real Time In POSIX

POSIX working group 1003.4 is attempting to standardize the various real-time extensions. Prior to participation in 1003.4 by the NGCR OSSWG and the VITA (ORKID standard) members, the working group activities were focused primarily on "soft" real-time issues. Now, these participants have joined with the real-time system vendors in ensuring that "hard" real-time is given its due. It is POSIX 1003.4 policy that its work will also address usability of the extensions for other than real-time systems whenever possible. The following enhancement categories are in progress:

1. Semaphores provide a facility for synchronization among multiple processes contending for access to a shared resource. The traditional UNIX approach (lock files) is too time consuming and disk intensive to be useful in high performance real-time systems, especially when expected contention for the resource is very low, as is typically the case.

2. Process memory locking provides an application API allowing the user to designate certain program and/or data memory to be excluded from the normal UNIX virtual memory management paging/swapping algorithms. This allows critical memory regions to be guaranteed prompt accessibility and minimizes nondeterministic behavior due to mass storage latency.

3. Shared memory interfaces enable a high bandwidth and high performance form of interprocess communication when the hardware supports this, the real-time constraints require this, and the protection afforded by more structured forms of IPC can be sacrificed.

4. Priority scheduling interfaces permit real-time applications to override the de-facto "time-sharing" UNIX style process scheduling policy with various priority based scheduling policies more appropriate to real-time multitasking. Only by doing this can hard real-time deadlines be guaranteed.

5. Realtime Signals extends the classic UNIX signal concept by allowing arbitrary user defined signals to be attached to user initiated actions and external events, and subsequently notifying the user process (synchronously or asynchronously) when the event is triggered.

6. Clocks and timers provide APIs to various resolution clocks and interval timers that provide better granularity and more flexibility than the traditional UNIX 1/Hz-second clock (time) and 1-second interval timer (alarm, sleep). Real-time systems usually have tight timing tolerances that are best met by millisecond or better-resolution low-jitter clocks and timers.

7. IPC Message passing addresses the need for a form of interprocess communication interface that is not inexorably tied to any specific implementation but that supports loosely coupled LAN-based communications typical among component subsystems of a large combat systems, as well as high performance shared memory based communications between cooperating processes in a uniprocessor or multiprocessor. The traditional UNIX IPC mechanisms (pipes, signals, and files) are often too restrictive or heavyweight for use in real-time systems.

8. Synchronized input and output provides interfaces whereby an application can guarantee that a set of data recorded in mass storage is current and self consistent. Traditional UNIX I/O assumes that the "OS knows best" but fails to address the need for embedded real-time systems to more closely control the reading and writing of data that might be needed for recovery purposes or might be written and read by different components of the system.

9. Asynchronous input and output provides alternative I/O interfaces that al[illegible]ngle process to initiate I/O to one or several devices simultaneously and continue processing w[illegible] awaiting I/O to complete. The traditional UNIX approach to this is to create separate processes to perform each I/O operation as well as queuing and notification functions. While this approach can actually yield more

structured programs, real-time systems often cannot tolerate the extra process context switching overhead.

10. *Advisory information interfaces provide additional information* to the OS file system so that the OS can optimize file access (reduce latency, prevent fragmentation, speed addressability) for real-time applications. This serves to improve performance and eliminate the non-determinism typically associated with UNIX file access.

11. POSIX threads provide a complete API set for lightweight processes that can coexist with the heavier POSIX process model. Threads within a single POSIX process share a considerable amount of state information (including memory); thus, context switching among threads experiences lower overhead, and interthread IPC can take advantage of the inherent shared memory. Additionally, threads provide a second level of concurrency model that matches quite nicely with the two levels implicit in the Ada programming language (several tightly coupled Ada tasks per Ada program, several loosely coupled Ada programs per system).

12. The Spawn process creation primitive provides an enhancement over the traditional 1003.1 fork() and exec() APIs for real-time systems. The 1003.1 interfaces imply not only the existence of a file system, but also a two step method of starting a new process which forces an often unnecessary duplication of an existing process. Spawn provides the more or less conventional real-time practice of "create process" with a single interface.

13. Timeouts for Blocking Services adds the conventional real-time capability of attaching an upper bound to the amount of time which several critical real-time interfaces may block a requesting process or thread. This capability is used primarily to increase the robustness of real-time applications in fault situations.

14. Execution Time Monitoring provides the ability for a process or thread to check the cumulative execution time of itself or another process or thread, and to establish CPU time limits. Such interfaces are essential in deadline driven real-time systems to ensure that all processes and/or threads are given fair opportunity to meet their deadlines.

15. Sporadic Server interfaces complement Priority Scheduling interfaces in real-time systems driven by external aperiodic requests. These simplify the schedulability analysis (as in rate monotonic scheduling theory) of such a real-time system because they allow aperiodic processes or threads to be treated as if they were periodic.

16. Device Control standardizes the format of interfaces to device drivers which go beyond the 1003.1 open/close/read/write/seek interfaces. Real-time systems typically utilize unique devices with unique "out-of-band" control requirements. UNIX has always provided an ioctl() interface to invoke such control actions as unloading a magnetic tape or setting the baud rate of a communication port. Device Control is a natural extension of these capabilities to general control requirements for arbitrary devices (such as radar or analog-to-digital converters). It does not attempt to define actual control requirements, only the interfaces necessary to pass control information.

17. Interrupt Control provides standard interfaces for connecting architecture and hardware dependent interrupts to application code. Real-time applications frequently need asynchronous notification of the occurrence of some hardware generated event. Performance is often an issue, so Interrupt Control addresses performance and other tradeoffs associated with different methods of asynchronous notification (Note that POSIX otherwise supports only a single method of asynchronous event notification, the Signal).

18. Typed Memory Allocation adds interfaces to POSIX which support dynamic memory allocation. POSIX had previously deferred all memory allocation interfaces to the ANSI C standard. Given the evolution of other languages which require dynamic memory allocation, and the proliferation of real-time systems which utilize several types or partitions of memory from which such allocation is possible, the ANSI C malloc() interface is no longer adequate.

4.2.2 Real Time in NGCR OS

Although POSIX interfaces differ substantially from most conventional real-time operating systems used heretofore in Navy systems, the substantial progress achieved by 1003.4 coupled with increased industry impetus toward real-time UNIX implementations would indicate that POSIX will eventually be an acceptable OS interface for all but the smallest and most time critical Navy applications.

Real-time profiles being developed by P1003.13 will stress the need for high performance OS implementations for real-time systems. The interfaces themselves cannot generally be evaluated with respect to performance because performance is a characteristic of an implementation, not an interface. However, performance metrics are being developed as part of the standards, and substantial effort has been expended to ensure that the real-time interfaces do not preclude efficient implementations. Thus, it is reasonable to expect that the Navy will be able to purchase good real-time operating system implementations compliant with the POSIX interface standards. This means that, in spite of the fact that POSIX interfaces are quite unlike those found in conventional real-time operating systems, NGCR OS based on POSIX will support real-time applications once real-time programmers understand and accept the POSIX-like interfaces.

4.2.3 NGCR/POSIX Real-Time Delta

The following unfulfilled requirements are especially significant to real-time applications because missing capabilities prevent a fine degree of control over the performance of the system in functions common to most real-time applications:

1.21 Bounded OS Service Times and Context Switching
6.3 File Management Scheduling
7.1 *Device Driver Availability*
16.10 Monitor Task's Execution Status (Ada)
16.17 Enable/Disable Interrupts (Ada)

The following unfulfilled requirements are especially significant to multiprocessor and distributed real-time systems because of the lack of a standardized approach to handling global time:

15.4 Selection of primary reference clock
15.5 Locate primary reference clock.

The following unfulfilled requirements are also significant to real-time systems but reflect capabilities which are not common to all real-time systems or are typically out of the mainstream of real-time processing.

1.17 Error conditions
1.23 Transaction scheduling information
5.1 Event and error receipt
5.2 Event and error distribution
5.3 Event and error management
5.4 Event and error logging
10.2 Execution history
11.* Reliability, adaptability, and maintainability (all)
13.5 Transaction scheduling information.

The following unfulfilled requirements may have some bearing on the performance of some real-time systems, although the relationship is a secondary one:

4.1 Data interchange services
9.2 Terminate Process

9.11 Examine process status
9.13 Save/restart process
12.3 Dynamic memory allocation and deallocation
12.4 Dynamic memory protection
12.5 Shared memory (as unfulfilled - for code segments)
16.6 Restart task (Ada)
16.15 Memory allocation and deallocation (Ada).

4.3 FAULT-TOLERANT SYSTEMS

Because many of the Navy systems to utilize NGCR OS will be mission critical, the OS must support the ability to detect, report, isolate, and recover from any foreseen hardware or software failure, thereby ensuring that the effects of such a failure on the mission are minimal. Fault tolerance requirements are explicitly seen in service classes 5 (event and error management) and 11 (reliability, adaptability, and maintainability), while some other requirements also have implications in this area.

4.3.1 Fault Tolerance in UNIX

Unfortunately, UNIX systems have traditionally had poor fault tolerance. Generally, software errors generated by an application and some hardware errors related to a device in use by an application are reported back to the application either synchronously (via error return codes and the "errno" system variable) or asynchronously (via a signal). The OS assumes no further role in the processing or logging of such errors, nor are there any services that assist in the recovery from errors. Furthermore, software errors detected within the UNIX kernel, and many hardware errors, cause the OS to simply give up. For example, many UNIX systems will not configure themselves around failed memory but instead inform an operator and halt, awaiting reboot of the system or they reboot themselves automatically (a process that takes from one to many minutes). In these cases, all user applications die in their tracks with no potential to recover anything unless the application has generated its own checkpoints. Curiously, in these circumstances, the error is frequently logged in a file accessible to the system administrator.

UNIX behaves this way because its typical users have been running applications in a time-sharing environment where centralized error handling and dynamic recovery are not the rule, but where having a system administrator in the loop is.

4.3.2 Fault Tolerance in POSIX

There had previously been little effort in the POSIX community to standardize fault tolerance related interfaces. This issue was generally considered out of scope. For example, a significant portion of the 1003.4 working group membership had been opposed to providing timeouts on blocking services because they can't imagine that software bugs end up in fielded systems. Recently, the hard real-time contingent of that working group has pushed for the kinds of fault tolerant capabilities that provide the characteristic robustness of mission-critical real-time systems.

OSSWG has led a Fault Management and Administration study group within POSIX over the past two years. While this group had initially confirmed that existing practice in fault tolerant operating systems is not mature enough to begin a standardization effort immediately, they have nonetheless brought this concern to the forefront. The group continues to work toward standardized Fault Management and Administration interfaces based on proposals by several industry groups including UNIX International and the Open Software Foundation.

4.3.3 Fault Tolerance in NGCR OS

NGCR OS requirements specify centralized facilities for receipt, coordination, distribution, delivery, and logging of error events, whether those events are detected by hardware or software, whether they indicate a hardware or software fault, and whether the fault occurs within the application or the operating system. The OS is expected to collect and retain as much information as possible about a fault that has occurred and provide access to this information to an application (not just a system administrator). This applies to faults detected asynchronously, as well as to faults discovered by application initiated hardware diagnostic tests. For transient faults, the OS must be configurable with thresholds that establish the tolerance level for errors. Isolation of faults to a system component must be supported, and the OS must be able to take predetermined actions based on fault severity. Ultimately, the OS must support reconfiguration of its own and application resources when one or several components of the system have failed, or upon application request.

4.3.4 NGCR/POSIX Fault Tolerance Delta

POSIX and UNIX compliant systems today provide virtually none of the required support. Although it is not required for many Navy systems, the NGCR OS interface will have to augment POSIX substantially to achieve a fault tolerance level acceptable to some mission critical Navy profiles. The most likely route to this goal is by closely following the activities of UNIX International, OSF, and X3T8 in these areas. As the concepts being explored by these groups become more well defined, either de-facto industry standards will emerge or it will become appropriate to reconsider introduction of a POSIX PAR to bring such de-facto fault tolerance interfaces into POSIX scope.

4.4 SECURITY

As stated in section 3.3, although P1003.1e and P1003.2c meet or support most of the OSSWG security requirements, further guidance is provided and required by the TCSEC and SECNAV Instruction 5239.2 "Information Security Instruction." The subject of the TCSEC and its interrelationship with the NGCR standards for security raises several issues:

1. The relationship between the P1003.1e/P1003.2c and the TCSEC standard.

2. The integration of common security-related features between various standards (e.g., NGCR, DoD, ISO) and which standard takes precedence.

3. The integration of common functions and features as the result of using two or more standards-based trusted commercial-off-the-shelf (COTS) products when they become available. This must also consider the integration of different TCSEC class COTS products or systems.

Navy acquisition programs must comply with DoD directives and the SECNAV instruction. Both recommend the TCSEC standard to develop security requirements for acquisition programs. The TCSEC is a collection of security criteria organized into classes. In most acquisitions, requirements may be specified from different TCSEC classes based on the criticality of the mission and the level of physical, procedural, operational, and communication security at the operational sites. For some specific acquisition programs or missions, the requirements cited for a particular TCSEC class may not all apply. NGCR OSSWG has reviewed P1003.1e and P1003.2c and found them compatible with the TCSEC criteria. [Note: In annex B of P1003.1e, the POSIX security subcommittee gives its reasons for choosing the TCSEC as the main source of security criteria.] As it defines each of the functions within each category of the interface standard (i.e., DAC, MAC, Privileges, Audit, Information Labels), P1003.1e and P1003.2c attempt to ensure that the security portion of the standards does not preclude meeting the higher class TCSEC systems. Although it is not explicitly cited in P1003.1e or P1003.2c, it is implied that to qualify as a TCSEC class system the P1003.1e and P1003.2c interface requirements must be developed in conjunction with the corresponding criteria stated in the TCSEC.

The integration of common security-related features between the various standards is non-trivial. Likewise, the use of trusted portable application software between systems built on different hardware platforms having a similar POSIX interface may require further examination of the application software. In either case when combinations of NGCR standards or standards-based COTS products are used, further system level analysis is required to identify, address, and resolve the significant integration issues.

An example which illustrates both issues addressed above is labeling. POSIX treats a label as an unstructured, undefined opaque object for portability purposes. This allows each vendor or developer of trusted application software who uses the P1003.1e and P1003.2c standards to define the internal structure of the label. From a standalone, homogeneous system perspective, this may not cause significant problems for Navy system engineers. However in a distributed, heterogeneous system when several NGCR standards and/or standards-based trusted application products are integrated, additional requirements may be necessary to define a common label format. This may be especially the case when trusted application programs are created to perform label transformations for mission-critical systems and such software must be totally correct. Such trusted application programs in general may not be transferable among heterogeneous POSIX-based systems.

The security requirements and the implementation of these requirements should always be viewed in terms of the TCSEC. P1003.1e and P1003.2c are interface standards that do not preclude meeting the TCSEC class requirements. However, P1003.1e and P1003.2c in themselves, being interface-related standards, cannot address all the operating system security requirements. The design and implementation of the P1003.1e and P1003.2c standards must be used in conjunction with requirements from the TCSEC classes to provide a well-defined system and a potentially certifiable secure product.

4.5 HETEROGENEITY

It has been a goal of the NGCR OS to support heterogeneous systems; that is, the same OS interface must not only support a variety of processor architectures, but it must allow dissimilar processors to cooperate as part of a larger system. This can take the form of heterogeneous processors on the same backplane (Futurebus+) or more commonly, heterogeneous processor types at different nodes of a distributed system.

4.5.1 Heterogeneity in UNIX

Today's UNIX systems support heterogeneity largely through the use of network services that provide commonality of function and information representation among different processor types (some running different vendors' UNIX) that share a common network medium and protocol (e.g., ethernet). Examples are network file system (NFS) and remote shell (rsh) capabilities. Such services typically do not attempt to solve data interchange format problems (word size, floating point format, endian-ness), leaving that as an exercise for the user; however, they do allow applications to work together fairly well in a heterogeneous distributed environment.

Few UNIX systems today support heterogeneity on the same backplane, simply because that is not a typical configuration. Notable exceptions such as Wind River's VxWorks do allow host (e.g., Sun workstation) and target (e.g., Mizar SPARC/VME-based real-time subsystem) to share a common backplane and memory.

4.5.2 Heterogeneity in POSIX

The POSIX standards effort is a giant step forward in supporting heterogeneity, since it attempts to standardize not only the basic interfaces (thus ensuring source code portability), but also the distributed services (thus allowing for universal interoperability, at least across a network). The issue of heterogeneity in a multiprocessor (dissimilar processors sharing memory) is not addressed by POSIX except in the distributed context.

4.5.3 Heterogeneity in NGCR OS

Heterogeneity is not called out in any specific NGCR OS requirement (though service class 4, data interchange interfaces, certainly hints at it). This is because the ability of one implementation of an operating system to work harmoniously with another implementation is largely an implementation issue. For example, if two implementations of a file system namespace use the standardized interface but two different character sets, then the ability to share namespace information between these implementations is severely hampered. The OSSWG should (1) attempt to identify those POSIX implementation dependencies that are detrimental to heterogeneity and (2) create an "implementor's guide" to promote increased interoperability.

4.5.4 NGCR/POSIX Heterogeneity Delta

Although the POSIX standardization effort and POSIX distribution standards are a strong positive step for heterogeneous systems, the POSIX motive is source code portability, not interoperability. Thus, it is unlikely that initial implementations of POSIX-compliant systems will work trouble-free in a heterogeneous environment. The POSIX (and thus, the NGCR OSSWG) focus on APIs simply does not address standardization of certain system interfaces (particularly OS-to-OS interfaces and global resource management).

4.6 ADA

The Ada programming language is not only the mandated DoD standard (and thus Navy standard) programming language, but is an international standard for large scale, long-lived, reliable applications. The Ada language is somewhat unique in that it defines within the language a number of operations that heretofore were considered to be in the domain of the target operating system, but that ultimately must be supported by an operating system component. Some Ada compilers are targeted to the bare machine; that is, the compiler vendor supplies the full underlying operating system. Other Ada compilers are targeted to a machine already running particular operating systems; in this case, the Ada vendor's run-time support package and/or the generated code itself interfaces with an operating system supplied by another vendor (typically, the computer vendor) whenever operating system services are required.

The Ada language also, like other language standards, specifies certain required library packages that must rely on operating system services for support.

Examples of operating system services implicit in the Ada language are the Ada tasking model (entry call, accept, select, etc.), the delay statement, the "new" allocator, and various Ada exceptions that may originate as machine-specific hardware interrupts (Numeric_Error, for example). Examples of Ada library packages that require operating system support are Text_IO, Low_Level_IO, IO_Exceptions, Unchecked_Deallocation, and Calendar.

4.6.1 Ada in UNIX

UNIX-based systems have been popular platforms for Ada language implementations, but there has been a great deal of misunderstanding and controversy surrounding such implementations. UNIX implementations have typically been a poor fit for the services required by the Ada language. For example, UNIX kernels have no schedulable entity that maps to an Ada task, so UNIX-based Ada implementations have usually provided a library level scheduler for Ada tasks. This approach has two drawbacks. First, whenever such an Ada task must invoke an operating system service that blocks, all the Ada tasks in the Ada program are blocked instead of only the one requiring the blocking service; second, the timely execution of the Ada tasks cannot be guaranteed because the UNIX process in which the Ada tasks live itself competes for the CPU via a different scheduler (the UNIX process scheduler). Another example of a poor fit is the various Ada timing services. Because UNIX provides timing services only at 1-second resolution, Ada implementations have been forced to use some fairly inaccurate and inefficient polling methods of timing. Even the Ada line and record-oriented I/O models are poorly supported by the UNIX byte-stream I/O model.

Generally, the outcome of this poor fit is that portable Ada programs don't work exactly as might be expected, either from the Ada perspective or from the UNIX perspective. Vendors, realizing this, typically provide additional nonstandard libraries to allow Ada programs to be more "UNIX like." Unfortunately, this does very little for portability, even from one Ada compiler implementation to another on the same UNIX operating system.

4.6.2 Ada in POSIX

POSIX has been supporting Ada through the 1003.5 working group, the product of which is to be a standard that makes the functionality of ISO/IEC 9945-1:1990 (1003.1) available to the Ada programmer. The P1003.20 working group is doing the same for the evolving real-time extensions (1003.4, 1003.4a, and 1003.4b).

It is important to note what 1003.5 does and does not attempt to do. In particular, 1003.5 provides an Ada language binding to POSIX interfaces; i.e., an Ada-like way to invoke POSIX services. It does NOT attempt to define POSIX interfaces suitable for supporting all the Ada run-time capabilities. Generally speaking, the POSIX community seems to feel that the latter is not in its scope. Nonetheless, recent activity in 1003.4 (i.e., concern that Pthreads be usable as Ada tasks) indicates that there is increasing sentiment toward supporting POSIX in the Ada run-time environment. The 1003.5 working group is currently debating the inclusion or exclusion of Ada bindings to real-time interfaces that would conflict with capabilities of the Ada run-time, or that would allow an Ada run-time environment to be written in Ada.

4.6.3 Ada in NGCR OS

It is essential that NGCR OS support not only an Ada language binding to all defined OS interfaces, but also the implicit interfaces required by the Ada run-time and the standard Ada library packages. These latter requirements are pretty much detailed in OSSWG requirements for service class 16, while the language binding requirements appear in service class 1.

In cases where an OSSWG requirement is satisfied directly within the language or from a standard Ada library package, and an explicit binding to the underlying service interface adds no functionality, the explicit binding is not necessary. For example, the POSIX "sleep" interface adds no functionality over and above the Ada "delay" statement and it is therefore unnecessary to have an Ada binding to the OS "sleep." Also, where an OS interface exists wholly to support a different language binding, an Ada binding makes no sense (e.g., the 1003.4a C interface "pthread_equal" exists because comparison of opaque types using the C operator == is invalid for pointer implementations of such types).

In support of the goals of application portability and reusability, NGCR applications must avoid the practice of substituting nonstandard language constructs and library packages for standard Ada capabilities. Toward this goal, it is essential that the NGCR OS implementations support standard Ada capabilities with very high performance, since performance requirements of real-time systems often take precedence over software engineering goals. Hopefully, as Ada matures into Ada-9X, new standard capabilities will be added to compensate for some of the architecture and OS dependent problems that have previously forced use of nonstandard interfaces.

4.6.4 NGCR/POSIX Ada Delta

The 1003.5 working group, in its process of drafting P1003.20, has started debating, and will continue to debate, such issues as providing Ada bindings to POSIX interfaces that duplicate or conflict with Ada run-time features, and providing support for Ada run-time environments written in Ada. Once such decisions have been made, the exact relationship between POSIX and Ada will be more well defined. POSIX 1003.1, 1003.4, P1003.4a, and P1003.4b certainly appear at this time to support Ada-83 fairly completely and, assuming no highly unusual policy is forthcoming from the 1003.5 working group, the delta appears small.

5. CONCLUSIONS

This document has carefully analyzed each NGCR OSSWG interface requirement (except for the very general requirements in Service Class 1) as it relates to the POSIX standardization effort. Of the 155 OSSWG requirements analyzed, 99 are directly met by the existing POSIX interfaces; Section 3 documents this mapping. Of the 56 remaining requirements, 44 have been classified as significant unfulfilled requirements. The remaining 12 have been, or are being re-evaluated or re-formulated in a manner more in keeping with a POSIX philosophy; these will ultimately either become met requirements, or be dropped entirely as OSSWG requirements.

The 44 significant unfulfilled requirements generally fall into one of three classifications: those that are nearly met by POSIX with the exception of minor details (13), those that clearly belong within the POSIX framework but have not yet been addressed by POSIX (14), and those which are outside the scope of the existing POSIX working groups (17). This "magnitude of delta" for each requirement is more significant than the actual count of unfulfilled requirements. When analyzed by service class, there are only a few trends (primarily the lack of POSIX support for service classes 5 and 11); but when the requirements are classified by importance to the "Big 6" technology areas, as is done in Section 4, the relative magnitudes of delta becomes clear: POSIX is moving in a positive direction in the areas of Real-Time Systems, Security, and Ada, with only follow-up work required to satisfy most related OSSWG requirements; while the POSIX framework currently addresses areas of Distributed Systems and Heterogeneity, there is substantial additional work required to bring these up to OSSWG standards; and finally the Fault Tolerance area has only recently been broached by POSIX through the OSSWG initiated Fault Management and Administration study group and its pending POSIX Services for Reliable, Available, and Serviceable Systems project.

In the strategy analyses of Section 3, it was found that many OSSWG requirements would be best met by working within the POSIX working groups and balloting groups to ensure that existing capabilities are extended or tuned, and that the necessary new capabilities are added; indeed this method has been in use since NGCR OSSWG became active in the POSIX activities, and substantial progress has already been observed, especially in the real time and networking areas. Over half of the significant unfulfilled requirements suggest this approach, and if the POSIX Services for Reliable, Available, and Serviceable Systems PAR is approved, virtually all of these requirements can be ultimately realized within the POSIX framework. Since it has always been an OSSWG goal for the OSIF to be fully under the purview of a single standards body, this is very encouraging progress indeed.

It was not always the case that this many requirements had a "home" within POSIX. First, OSSWG initiated the Real Time Distributed Systems Communication (1003.21) project which has completed its requirements analysis process and has begun drafting a standard ich will meet most of the unfulfilled OSSWG networking requirements. Second, a Distributed Security (1003.22) project was approved to address the unfulfilled security requirements and how the POSIX security interfaces will support distributed systems. Third, although the Fault Management and Administration Study Group had concluded that it was inappropriate for POSIX to standardize on Fault Tolerance interfaces a year ago, that OSSWG initiated group continues to gather industry support, has closely followed the evolution of various non-POSIX efforts in the Fault Tolerance arena (e.g. UNIX International, OSF, X3T8), and expects to become a fully recognized POSIX project in the near future. These three relatively new efforts have provided the foundation for many of the most difficult delta resolutions. Finally, recommendations have been made to attach to other existing and evolving standards outside the POSIX framework where appropriate (e.g. P1256/OBIOS, ANSI/RPS, ASN.1, Network Time Protocol, and the Dwight Wilcox Distributed Realtime Clock Synchronization approach), but only if and when OSSWG has exhausted all POSIX resolution alternatives.

This document defines much of the remaining work ahead for the NGCR OSSWG, especially as its members debate and ballot the various existing POSIX draft standards and contribute to new ones. It also serves as an important basis for the ultimate product of the NGCR OSSWG, a military handbook or technical specification for the NGCR OSIF.

This document recommends a number of OSSWG requirements be re-evaluated. Changes to the requirements in the OCD may result from this re-evaluation process, which in turn may cause some change of deltas in the next Delta Document revision. In addition to specifically recommended re-evaluations, OSSWG has two general concerns about the requirements: First, OSSWG had purposely avoided addressing the semantics of each requirement in a distributed computing environment because of the relative immaturity of distributed services within POSIX; that area has matured substantially, and the time has come to explicitly split each requirement (where it makes sense) into its non-distributed and its distributed context. Second, certain requirements have been perceived as dictating an implementation to meet a requirement rather than stating the true requirement and giving the operating system the freedom to meet it in the best way possible in a given implementation; requirements such as Write Contiguous File (6.20) and Unacknowledged Connection Oriented Service (8.4) are examples. Therefore, OSSWG recommends a thorough review of the OCD requirements, addressing these two overall concerns, prior to the next Delta Document revision.

This Version 4 of the Delta Document is not the final version. This is a living document and will change as (1) POSIX evolves, and (2) the OSSWG is able to develop new methods of satisfying the remaining deltas. We intend to update this document yearly, at least until completion of the military handbook or technical specification.

APPENDIX A

DELTA SUMMARY AND CROSS REFERENCES

The table on the following pages lists, for each OSSWG requirement (except the general requirements of service class 1) references to all POSIX interfaces which OSSWG believes fully or partially fulfill the requirement. The POSIX document number, paragraph number(s), and a brief description of the pertinent interfaces and/or capabilities is provided.

In addition, each unfulfilled requirement is coded with a rating indicating its significance to the overall NGCR OS interface standardization effort: A rating of "a" indicates that standardization of interfaces which meet the requirement is essential; a rating of "b" indicates that standardization of interfaces which meet the requirement is highly desirable; a rating of "c" indicates that fulfilling the requirement can be deferred to a later date; a rating of "d" indicates that the OSSWG should re-evaluate the need for standardized interfaces fulfilling the requirement.

Finally, for each unfulfilled requirement, the OSSWG recommendations are summarized.

DELTA SUMMARY AND CROSS REFERENCES

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
2.1 Non-NGCR System Interface	1003.1	3.1-3.4	Process model and interprocess signals	-	No Delta
		6.1-6.5	Device input/output and interprocess pipes		
	1003.4	3.3	Realtime interprocess signals		
		6.6	Synchronized files on shared file systems		
		6.7	Asynchronous input/output		
		12.3	Shared memory		
		15.1-15.3	Message passing		
	1003.5	3.1-3.3	Process model and interprocess signals		
		6.1-6.2	Device input/output and interprocess pipes		
	P1003.12	4	Detailed Network Interface		
	P1003.20	3.1	Realtime interprocess signals		
		6.6	Synchronized files on shared file systems		
		6.7	Asynchronous input/output		
		12.3	Shared memory		
		15.1-15.2	Message passing		
3.1 Audit Data Storage	P1003.1e	24.1.2	Audit records	-	No Delta
3.2 Audit Generation	P1003.1e	24.1.2	Audit records	-	No Delta
		24.1.3	Audit interfaces		
3.3 Audit Record Contents	P1003.1e	24.2.2	Audit ... record contents	-	No Delta
3.4 Audit Data Manipulation	P1003.1e	24.4ff.	Audit functions	-	No Delta
3.5 Device Labels	P1003.1e	27.3.1	Initial info label	-	No Delta
3.6 Basic DAC	P1003.1e	23.1.5	ACL access check algorithm	-	No Delta
3.7 DAC Inclusion/Exclusion	P1003.1e	23.4.1	Add a permission	-	No Delta
		23.4.8	Create a new ACL entry		
		23.4.11	Delete an ACL entry		
		23.4.12	Delete permissions		
3.8 DAC Propagation	1003.1	5.6.4	Change file modes	-	No Delta
	P1003.1e	23.1.2	Rel. w/ file perm bits		
3.9 Labeling of Export Channels	P1003.1e	27.1.2ff.	Info label policy	-	No Delta
		27.3.4	Floating info labels		
3.10 Setting Communication Labels	P1003.1e	27.3.12	Set process info label	-	No Delta
3.11 Identification and Authentication	P1003.1e	B.1.3.7.1	Deferred w/ expl	-	No Delta
	1003.1	4.2	User identification		
		9.2	User and group database		
	1003.5	4.1.3-4.1.4	User identification		
		9.1-9.2	User and group data base		
3.12 Labeling of Human Readable Output	P1003.1e	27.3.14	Convert int label to text	-	No Delta
3.13 Subject and Object Labeling	P1003.1e	26.3.10-12	Set the label ...	-	No Delta
3.14 Label Contents	P1003.1e	27.3.10-12	Set info label ...	-	No Delta
3.15 MAC Policy	P1003.1e	26.1.2	MAC policy	-	No Delta
3.16 MAC Manipulation	P1003.1e	26.3.10-12	Set the label ...	-	No Delta
		27.3.10-12	Set info label ...		
	P1003.2c	11.3	setfmac - set the MAC ...		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
3.17 Object Reuse	P1003.1e	B.1.3.7.3	Implementation issue not appropriate to standard. A conforming implementation may implement a strong *object reuse policy without impacting the API.*	d	No further action required
3.18 User Notification of Sensitivity Label	P1003.1e	B.1.3.7.3	Deferred with explanation	-	No Delta
3.19 Sensitivity Label Query	P1003.1e	B.1.3.7.3	Deferred with explanation	-	No Delta
3.20 System Integrity	P1003.1e	B.1.3.7.6.3	Deferred with explanation	-	No Delta
3.21 Identification of Users Based on Roles	P1003.1e	B.1.3.7.1	Deferred with explanation	-	No Delta
3.22 Least Privilege	P1003.1e	25.1ff. 25.4.15	General overview: capabilities SetProcsCapabilityState	-	No Delta
3.23 Trusted Path	P1003.1e	B.1.3.7.6.8	Beyond scope due to POSIX.1 definition of "terminal" limited to dumb ttys.	d	No further action required
3.24 Trusted Recovery	P1003.1e	B.1.3.7.6.6	Out of scope: "not a subject for application portability"	d	No further action required
4.1 Data Interchange Service (Data Format Conversion	P1351 P1353	Unknown Unknown	API for ASN.1 API for ASN.1	a	Add floating point to P1351/P1353 Support standardization of XDR
5.1 Event and Error Receipt	1003.1 1003.4 P1003.4a P1003.4b 1003.5 P1003.20	2.4 3.3 7.1.1.9 3.3 3.3 22.3 2.4.4 3.3 7.1.1.9 3.1	Errors reported via error return code or errno Signals generated as result of terminal I/O Signals generated as result of terminal I/O *Realtime queued signals* Signals as extended to threads environment Interrupt capture, locking, notification Error codes and exceptions Signals generated as result of terminal I/O Signals generated as result of terminal I/O Realtime queued signals	c	UNIX International, OSF, POSIX SRASS group, and X3T8
5.2 Event and Error Distribution	1003.1 1003.4 P1003.4a P1003.4b 1003.5 P1003.20	2.4 3.3 7.1.1.9 3.3 3.3 22.3 2.4.4 3.3 7.1.1.9 3.1	Errors reported via error return code or errno Signals generated as result of terminal I/O Signals generated as result of terminal I/O Realtime queued signals Signals as extended to threads environment Interrupt capture, locking, notification Error codes and exceptions Signals generated as result of terminal I/O Signals generated as result of terminal I/O Realtime queued signals	a	Support P1003.4b Interrupt Control (for events) UNIX International, OSF, POSIX SRASS group, and X3T8
5.3 Event and Error Management	1003.1 1003.4 P1003.4a P1003.4b 1003.5 P1003.20	2.4 3.3 3.3 3.3 22.3 2.4.4 3.3 3.1	Errors reported via error return code or errno Signals generated as the result of errors Realtime queued signals Signals as extended to threads environment Interrupt capture, locking, notification Error codes and exceptions Signals generated as the result of errors Realtime queued signals	a	Support P1003.4b Interrupt Control *(for events)* UNIX International, OSF, POSIX SRASS group, and X3T8

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
5.4 Event Logging	None			a	UNIX International, OSF, POSIX SRASS group, and X3T8
5.5 Enable/Disable Interrupts	P1003.4b	22.3	Interrupt control functions	-	No Delta
5.6 Mask/Unmask Interrupts	1003.1	3.3.5	Examine and change blocked signals	c	Inappropriate for standardization
	1003.5	3.3.7	Examine and change blocked signals		
6.1 Contiguous Read of a File	1003.1	6.4.1	Read from a file	-	No Delta
		6.7.4	List directed I/O		
	1003.4	6.7.2	Asynchronous read		
		6.7.4	List directed I/O		
		12.2.1	Memory mapping of files		
	P1003.4b	23.1.1	File advisory information		
	1003.5	6.1.2	Read and generic read from a file		
	P1003.20	6.7.2	Asynchronous read		
		6.7.4	List directed I/O		
		12.2	Memory mapping of files		
6.2 Protect an Area Within a File	1003.1	5.3.1	Open a file - access specification and denial	-	No Delta
		6.5.2	File control - advisory record locking		
	1003.4	12.2.3	Change protection of memory mapped file segment		
	1003.5	5.1.1	File permissions		
		6.2.1	Advisory record locking		
	P1003.20	12.2.3	Change protection of memory mapped file segment		
6.3 File Management Scheduling	None			c	Inappropriate for standardization
6.4 File Management Suspend/Resume for Processes	1003.1	5.3.1	Open a file with blocking/non-blocking behavior	-	No Delta
		6.4.1	Read from a file (blocking or non-blocking)		
		6.4.2	Write to a file (blocking or non-blocking)		
	1003.4	6.7	Asynchronous input/output		
	P1003.4b	21.1-21.3	Device control		
	1003.5	6.1.1	Open a file with blocking/non-blocking behavior		
		6.1.2	Read or write a file (blocking or non-blocking)		
	P1003.20	6.7	Asynchronous input/output		
6.5 File Management Block Requests	1003.1	6.4.1-6.4.2	Read from and write to a file	-	No Delta
		6.5.3	Reposition read/write file offset		
	1003.4	12.2.1	Memory mapping of files		
	P1003.4b	23.1.1	File advisory information		
	1003.5	6.1.2	Read from and write to a file		
		6.1.3	Reposition read/write file offset		
	P1003.20	12.2	Memory mapping of files		
6.6 Round Robin File Management			THIS REQUIREMENT HAS BEEN DELETED		
6.7 Open a File	1003.1	5.3.1-5.3.2	Open and/or create a file	-	No Delta
	1003.4	5.3.1	Open a file		
	P1003.4b	23.1.1	File advisory information		
	1003.5	6.1.1	Open and/or create a file		
	P1003.20	6.5	Open a file		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
6.8 Point Within a File	1003.1	6.5.3	Reposition read/write file offset	-	No Delta
	1003.4	12.2.1	Memory mapping of files		
	1003.5	6.1.3	Reposition read/write file offset		
	P1003.20	12.2	Memory mapping of files		
6.9 Read a File	1003.1	6.4.1	Read from a file	-	No Delta
	1003.4	6.7.2	Asynchronous read		
		6.7.4	List directed I/O		
		12.2.1	Memory mapping of files		
	1003.5	6.1.2	Read and generic read from a file		
	P1003.20	6.7.2	Asynchronous read		
		6.7.4	List directed I/O		
		12.2	Memory mapping of files		
6.10 Close a File	1003.1	6.3.1	Close a file	-	No Delta
	1003.4	6.3.1	Close a file		
	1003.5	6.1.1	Close a file		
6.11 Delete a File	1003.1	5.5.1	Remove a file's directory entry	-	No Delta
	1003.2	4.52 (Vol 1)	Remove files		
	1003.5	5.2.1	Remove a file's directory entry		
6.12 Create a Directory	1003.1	5.4.1	Make a directory	-	No Delta
	1003.2	4.41 (Vol 1)	Make a directory		
	1003.5	5.2.1	Make a directory		
6.13 Specify Default Directory	1003.1	5.2.1	Change current working directory	-	No Delta
	1003.2	4.5 (Vol 1)	Change current working directory		
	1003.5	4.3.3	Change current working directory		
6.14 Delete a Directory	1003.1	5.5.2	Remove a directory	-	No Delta
	1003.2	4.54 (Vol 1)	Remove a directory or directory tree		
	1003.5	5.2.1	Remove a directory		
6.15 Shadow Files	1003.1	5.3.1-5.3.2	Open and/or create a file	-	No Delta
		6.4.2	Write a file		
	1003.4	11.1-11.3	Semaphores for resource locking		
		12.2.1	Memory mapping of files		
	P1003.4a	11.2-11.3	Semaphores and mutexes for resource locking		
	1003.5	6.1.1	Open and/or create a file		
		6.1.2	Write or generic write to a file		
	P1003.20	11.1-11.3	Semaphores for resource locking		
		12.2	Memory mapping of files		
6.16 Create a File	1003.1	5.3.1-5.3.2	Open and/or create a file	-	No Delta
	1003.5	6.1.1	Open and/or create a file		
6.17 Query File Attributes	1003.1	5.6.1-5.6.2	Get file characteristics/status	-	No Delta
		5.6.3	Check file accessibility		
		6.5.3	Obtain file's read/write position and/or length		
	1003.5	5.3	Get file characteristics/status		
		5.2.6	Check file accessibility		
		6.1.3	Obtain file's read/write position and/or length		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
6.18 Modify File Attributes	1003.1	5.6.4-5.6.6	Change file modes, owner, group, times	-	No Delta
		6.5.3	Change file read/write position		
	1003.2	4.7 (Vol 1)	Change file modes		
	1003.4	5.6.7	Set the length of a file		
	P1003.4b	23.1.1	Set file advisory information		
		23.1.3	Modify file allocation		
	1003.5	5.2.5	Change file modes, owner, group, times		
		6.1.3	Change file read/write position		
	P1003.20	5.6.7	Set the length of a file		
6.19 Write a File	1003.1	6.4.2	Write to a file	-	No Delta
	1003.4	6.7.3	Asynchronous write		
		6.7.4	List directed I/O		
		12.2.1	Memory mapping of files		
	1003.5	6.1.2	Write or generic write to a file		
	P1003.20	6.7.3	Asynchronous write		
		6.7.4	List directed I/O		
		12.2	Memory mapping of files		
6.20 Write Contiguous File	1003.1	6.4.2	Write a file	-	No Delta
	1003.4	6.7.3	Asynchronous write		
		6.7.4	List directed I/O		
		12.2.1	Memory mapping of files		
	P1003.4b	23.1.1	File advisory information		
		23.1.3	File space control		
	1003.5	6.1.2	Write or generic write to a file		
	P1003.20	6.7.3	Asynchronous write		
		6.7.4	List directed I/O		
		12.2	Memory mapping of files		
7.1 Device Driver Availability	1003.4	12.2.1	Memory mapping of special devices	a	P1003.7
	P1003.4b	22.3	Interrupt control		
	P1003.7	6	Device class		
	P1003.20	12.2	Memory mapping of special devices		
7.2 Open Device	1003.1	5.3	General file creation	-	No Delta
	1003.5	5.2.1	Creating and removing files		
		6.1.1	Open or create a file		
7.3 Close Device	1003.1	6.3.1	Close a file	-	No Delta
	1003.5	6.1.1	Close a file		
7.4 Transmit Data	1003.1	6.4.2	Write data to a device	-	No Delta
	1003.4	6.7.3	Asynchronous write to device		
		6.7.4	List directed I/O to device		
		12.2.1	Memory mapping of special files (devices)		
	1003.5	6.1.2	Write and generic write		
	P1003.20	6.7.3	Asynchronous write to device		
		6.7.4	List directed I/O to device		
		12.2	Memory mapping of special files (devices)		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
7.5 Receive Data	1003.1	6.4.1	Read from a device	-	No Delta
	1003.4	6.7.2	Asynchronous read from a device		
		6.7.4	List directed I/O from a device		
		12.2.1	Memory mapping of special files (devices)		
	1003.5	6.1.2	Read and generic read		
	P1003.20	6.7.2	Asynchronous read from a device		
		6.7.4	List directed I/O from a device		
		12.2	Memory mapping of special files (devices)		
7.6 Device Event Notification	See 5.*			a	Refer to 5.*, Event and Error interfaces
7.7 Control Device	1003.1	6.5	Control operations on files	-	No Delta
		7.1-7.2	General terminal interface		
	P1003.4b	21.2	Device control		
	1003.5	6.1.5	Control operations on files		
		7.1-7.2	General terminal interface		
7.8 I/O Directory Services	1003.1	5	Files and Directories	-	No Delta
	1003.1a	11.1	File Hierarchy Streams		
	1003.5	5.2-5.3	POSIX_Files & POSIX_File_Status		
7.9 Device Management Suspend/Resume for Processes	1003.1	5.3.1	Open device with blocking/non-blocking behavior	-	No Delta
		6.4.1	Read from a device (blocking or non-blocking)		
		6.4.2	Write to a device (blocking or non-blocking)		
	1003.4	6.7	Asynchronous input/output to/from device		
	P1003.4b	21.1-21.3	Device control		
	1003.5	6.1.1	Open device with blocking/non-blocking behavior		
		6.1.2	Read, write, generic read/write		
	P1003.20	6.7	Asynchronous input/output to/from device		
7.10 Mount/Dismount Device	P1003.7	6	Device class	a	P1003.7
7.11 Initialize/Purge Device	1003.1	7.1-7.2	General terminal interface	-	No Delta
	P1003.4b	21.2	Device control		
	1003.5	7.1-7.2	General terminal interface		
8.1 Interface to NAVY Standard Network	P1003.12	4	Detailed Network Interface	a	New PARs - ROSE and Network Management
	1224	6	Service Interfase (for ASN.1 package)		
	1224.1	4	Message Handling Interface		
	1224.2	4	Directory Service Interface		
	P1238.1	Unknown	FTAM Service Interface		
	P1351	5	ACSE Service Interface		
8.2 Interfaces to Other Network and Communication Entities	P1003.12	4	Detailed network interface	-	No Delta
	1003.1	6.1-6.5	Input and output primitives		
	1003.5	6.1-6.2	Input and output primitives		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
8.3 Acknowledged Connection-Oriented Service	P1003.12	4.6.9	Receive	-	No Delta
		4.6.10	Send		
		4.6.12	Connect		
		4.6.17	Disconnect		
		4.6	Supporting functions		
		B.3.1.2.2	Quality of service options		
		B.3.1.3.2	Quality of service options		
	P1351	5.3.2	Close instance/disconnect		
		5.3.9	Open instance/connect		
		5.3.11	Receive		
		5.3.15	Send		
		5.3	Supporting functions		
8.4 Unacknowledged Connection-Oriented Service	P1003.12	4.6.9	Receive	-	No Delta
		4.6.10	Send		
		4.6.12	Connect		
		4.6.17	Disconnect		
		4.6	Supporting functions		
		B.3.1.2.2	Quality of service options		
		B.3.1.3.2	Quality of service options		
	P1351	5.3.2	Close instance/disconnect		
		5.3.9	Open instance/connect		
		5.3.11	Receive		
		5.3.15	Send		
		5.3	Supporting functions		
8.5 Acknowledged Datagram Service	P1003.12	4.6.4	Error	-	No Delta
		4.6.5	Init		
		4.6.6	Receive		
		4.6.7	Send		
		4.6	Supporting functions		
8.6 Datagram Transfer Service	1003.4	15.2	Message passing	-	No Delta
	P1003.4b	15.2	Message passing with timeouts		
	P1003.12	4.6.4	Error		
		4.6.5	Init		
		4.6.6	Receive		
		4.6.7	Send		
		4.6	Supporting functions		
	P1003.20	15.2	Message passing		
8.7 Request - Reply Service	None			a	1003.21 group; 1003.12 as backup
8.8 Broadcast/Multicast Service	None			a	1003.21 group; 1003.12 as backup
8.9 K- Acknowledged Multicast Service	None			a	1003.21 group; 1003.12 as backup
8.10 Atomic Multicast Service	None			a	1003.21 group; 1003.12 as backup

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
9.1 Create Process	1003.1	3.1.1-3.1.2	Create a process and execute a file	-	No Delta
		3.3.7	Wait for a signal (wait for start)		
		3.4.2	Wait for a signal (wait for start)		
		6.4.1	Read a message from a pipe (wait for start)		
	1003.1a	13.1	System function "system()"		
	1003.2	B.3.1 (Vol 1)	System function "system()"		
	1003.4	3.3.8	Wait for a signal (wait for start)		
		11.2.6	Wait for a locked semaphore (wait for start)		
		13.3.1	Set process scheduling parameters		
		13.3.3	Set process scheduling policy and parameters		
		15.2.5	Receive a message (wait for start)		
	P1003.4a	3.3.10	Wait for a signal (wait for start)		
		11.4.4	Wait on a condition (wait for start)		
		13.5.1-13.5.2	Setting thread scheduling attributes		
		16.1.1-16.1.2	Thread creation with attributes		
	P1003.4b	3.1.3	Spawn a process		
	1003.5	3.1.1-3.1.2	Create a process and execute a file		
		3.2.1-3.2.2	Create a process and execute a file		
		6.1.2	Read a message from a pipe (wait for start)		
	P1003.20	3.1.3	Wait for a signal (wait for start)		
		11.2.6	Wait for a locked semaphore (wait for start)		
		11.4.4	Wait on a condition (wait for start)		
		13.3.1	Set process scheduling parameters		
		13.3.3	Set process scheduling policy and parameters		
		13.5.1-13.5.2	Setting thread scheduling attributes		
		15.2.5	Receive a message (wait for start)		
9.2 Terminate Process	1003.1	3.2.2	Terminate a process (self)	a	P1003.4d
		3.3.2	Send a termination signal to a process		
	1003.2	4.32 (Vol 1)	Send a termination signal to a process		
	P1003.4a	16.1.4	Terminate a thread (self)		
		18.2.1	Cancel execution of a thread		
	1003.5	3.1.3	Terminate a process (self)		
		3.3.6	Send a termination signal to a process		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
9.3 Start Process	1003.1	3.3.2	Send a signal to a process (waiting to start)	-	No Delta
		6.4.2	Write a message to a pipe (waited for)		
	1003.1a	13.1	System function "system()"		
	1003.2	B.3.1 (Vol 1)	System function "system()"		
	1003.4	3.3.9	Send a signal to a process (waiting to start)		
		11.2.7	Unlock a semaphore (waited for)		
		15.2.4	Send a message (waited for)		
	P1003.4a	3.3.11	Send a signal to a thread (waiting to start)		
		11.4.3	Broadcast or signal a condition (waited for)		
	1003.5	3.3.6	Send a signal to a process (waiting to start)		
		6.1.2	Write a message to a pipe (waited for)		
	P1003.20	3.1.4	Send a signal to a process (waiting to start)		
		11.2.7	Unlock a semaphore (waited for)		
		11.4.7	Broadcast or signal a condition (waited for)		
		15.2.4	Send a message (waited for)		
9.4 Stop Process (See also 9.8 IPC)	1003.1	3.3.7	Wait for a signal (to resume)	-	No Delta
		3.4.2	Wait for a signal (to resume)		
		6.4.1	Read a message from a pipe (to resume)		
	1003.4	3.3.8	Wait for a signal (to resume)		
		11.2.6	Wait for a locked semaphore (until resumed)		
		15.2.5	Receive a message (to resume)		
	P1003.4a	3.3.10	Wait for a signal (to resume)		
		11.4.4	Wait on a condition (to resume)		
	1003.5	6.1.2	Read a message from a pipe (to resume)		
	ISO 9899	goto	Local jump to restart point		
		setjmp()	Define non-local restart point		
		longjmp()	Non-local jump to restart point		
	P1003.20	3.1.3	Wait for a signal (to resume)		
		11.2.6	Wait for a locked semaphore (until resumed)		
		11.4.8	Wait on a condition (to resume)		
		15.2.5	Receive a message (to resume)		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
9.5 Suspend Process (See also 9.8 IPC)	1003.1	3.3.7	Wait for a signal (to resume)	-	No Delta
		3.4.2	Wait for a signal (to resume)		
		3.4.3	Wait until time elapses		
		6.4.1	Read a message from pipe (to resume)		
	1003.4	3.3.8	Wait for a signal (to resume)		
		11.2.6	Wait for a locked semaphore (until resumed)		
		14.2.5	Wait until time elapses		
		15.2.5	Receive a message (to resume)		
	P1003.4a	3.3.10	Wait for a signal (to resume)		
		11.4.4	Wait on a condition (to resume)		
	1003.5	6.1.2	Read a message from a pipe (to resume)		
	Ada LRM	9.6	Wait until time elapses (delay)		
	P1003.20	3.1.3	Wait for a signal (to resume)		
		11.2.6	Wait for a locked semaphore (until resumed)		
		11.4.8	Wait on a condition (to resume)		
		14.2.5	Wait until time elapses		
		15.2.5	Receive a message (to resume)		
9.6 Resume Process	1003.1	3.3.2	Send a signal to a process (waiting to resume)	-	No Delta
		3.4.1	Schedule a signal to this process in the future		
		6.4.2	Write a message to pipe (awaiting resume)		
	1003.4	3.3.9	Send a signal to a process (waiting to resume)		
		11.2.7	Unlock a semaphore (waited for)		
		14.2.4	Send a signal to a process in the future		
		15.2.4	Send a message (waited for)		
	P1003.4a	3.3.11	Send a signal to a thread (waiting to resume)		
		11.4.3	Broadcast or signal a condition (waited for)		
	1003.5	3.3.6	Send a signal to a process (waiting to resume)		
		6.1.2	Write a message to a pipe (awaiting resume)		
	P1003.20	3.1.4	Send a signal to a process (waiting to resume)		
		11.2.7	Unlock a semaphore (waited for)		
		11.4.7	Broadcast or signal a condition (waited for)		
		14.2.4	Send a signal to a process in the future		
		15.2.4	Send a message (waited for)		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
9.7 Delay Process (See also 9.8 IPC)	1003.1	3.3.7	Wait for a signal (to end delay)	-	No Delta
		3.4.1	Schedule a signal to this process in the future		
		3.4.2	Wait for a signal (to end delay)		
		3.4.3	Wait until time elapses		
		6.4.1	Read a message from pipe (to end delay)		
	1003.2	4.57 (Vol 1)	Sleep utility "sleep"		
	1003.4	3.3.8	Wait for a signal (to end delay)		
		14.2.4	Send a signal to a process in the future		
		14.2.5	Wait until time elapses		
	P1003.4a	3.3.10	Wait for a signal (to end delay)		
	1003.5	6.1.2	Read a message from a pipe (to resume)		
	Ada LRM	9.6	Waite until time elapses (delay)		
	P1003.20	3.1.3	Wait for a signal (to end delay)		
		14.2.4	Send a signal to a process in the future		
		14.2.5	Wait until time elapses		
9.8 Interprocess Communication	1003.1	3.3	Signals	-	No Delta
		3.4.2	Signals		
		6.1.1	Pipes		
		6.4.1-6.4.2	Read and write pipes, files, sockets, etc.		
	1003.4	3.3	Signals		
		6.6	Synchronized I/O (file IPC)		
		6.7	Asynchronous I/O (file, pipe, socket IPC)		
		11.2	Semaphores		
		12.3	Shared memory		
		15.2	Message passing		
	P1003.4a	3.3	Signals		
		6.4.1-6.4.2	Read and write pipes, files, sockets, etc.		
		6.7	Asynchronous I/O (file, pipe, socket IPC)		
		11.2-11.4	Semaphores, mutexes, and condition variables		
		15.2	Message passing		
	P1003.4b	11.2.6	Semaphores with timeouts		
		11.3.3	Mutexes with timeouts		
		15.2	message passing with timeouts		
	1003.5	3.3	Signals		
		6.1.1	Pipes		
		6.1.2	Read and write pipes, files, sockets, etc.		
	P1003.12	4.6	FSM event operations		
	P1003.20	3.1	Signals		
		6.6	Synchronized I/O (file IPC)		
		6.7	Asynchronous I/O (file, pipe, socket IPC)		
		11.2-11.4	Semaphores, mutexes, and condition variables		
		12.3	Shared memory		
		15.2	Message passing		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
9.9 Examine Process Attributes (See also 9.8 IPC)	1003.1	4.1.1	Get my process and parent process IDs	-	No Delta
		4.2.1	Get my real and effective user and group IDs		
		4.2.3	Get my supplementary group IDs		
		4.2.4	Get my user name		
		4.3.1	Get my process group ID		
		4.5.2	Get my process times		
		4.7.1	Get my controlling terminal ID		
	1003.2	5.23 (Vol 1)	Process status utility "ps"		
	1003.4	13.3.2	Get scheduling parameters		
		13.3.4	Get scheduling policy		
		13.3.6	Get scheduling parameter limits		
	P1003.4a	13.5.2	Get thread scheduling parameters		
		16.1.5	Get my thread ID		
	P1003.4b	20.1.3	Access process CPU clock		
		20.1.5	Access process CPU clock attributes		
		20.1.6	Access thread CPU clock		
	1003.5	4.1.1	Get my process and parent process IDs		
		4.1.2	Get my process group ID		
		4.1.3-4.1.4	Get my real and effective user and group IDs		
		4.2.1	Get my process times		
		7.2.12	Get my controlling terminal ID		
	P1003.20	13.2	Get thread scheduling parameters		
		13.3.2	Get scheduling parameters		
		13.3.4	Get scheduling policy		
		13.3.6	Get scheduling parameter limits		
9.10 Modify Process Attributes (See also 9.8 IPC)	1003.1	4.2.2	Set my effective user and group IDs	-	No Delta
		4.3.2	Set my session ID		
		4.3.3	Set my process group ID		
	1003.4	13.3.1	Set scheduling parameters		
		13.3.3	Set scheduling parameters and policy		
	P1003.4a	13.5.2	Set thread scheduling parameters		
		18.2.2	Set thread cancelability state and type		
	P1003.4b	13.5.2	Set thread scheduling parameters		
	1003.5	4.1.2	Set my process group ID		
		4.1.2	Set my session ID		
		4.1.3-4.1.4	Set my real and effective user and group IDs		
	P1003.20	13.2	Set thread scheduling parameters		
		13.3.1	Set scheduling parameters		
		13.3.3	Set scheduling parameters and policy		
9.11 Examine Process Status	1003.1	3.2.1	Return process termination status	a	1003.7 and 1003.2 groups
	1003.2	5.23 (Vol 1)	Process status utility "ps"		
	P1003.4a	16.1.3	Return thread termination status		
	1003.5	3.1.5	Return process termination status		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
9.12 Process (Thread) Identification (See also 9.8 IPC)	1003.1	2.2.2.67	Definition of process ID	-	No Delta
		3.1.1	Obtain process ID of a new process (forked)		
		3.2.1	*Controlled reuse of process IDs*		
		4.1.1	Get my process and parent process IDs		
	P1003.4a	2.2.2.162	Definition of thread ID		
		16.1.2	Obtain thread ID of a new thread		
		16.1.3	Controlled reuse of thread IDs		
		16.1.5	Get my thread ID		
	P1003.4b	3.1.3	Obtain process ID of a new process (spawned)		
	1003.5	2.2.2.73	Definition of process ID		
		3.1.2	Obtain process ID of a new process		
		3.1.5	Controlled reuse of process IDs		
		3.2.1	Obtain process ID of a new process (forked)		
		4.1.1	Get my process and parent process IDs		
9.13 Save/Restart Process	1003.1a	11.2	Process checkpoint and restart capability	a	Investigate checkpoint/restart of thread in 1003.7 group
9.14 Program Management Function (See also 9.8 IPC)	1003.1	2.2.2.62	Definition of a process	-	No Delta
		3.1-3.2	Process creation, execution, termination		
	1003.4	13	Execution scheduling for processes		
	P1003.4a	*2.2.2.161*	*Definition of a thread*		
		13	Execution scheduling for threads		
		16.1	Thread creation, execution, termination		
	1003.5	2.2.2.68	Definition of process		
		3.1-3.2	Process creation, execution, termination		
	P1003.20	13	Execution scheduling		
10.1 Debug Support	1003.1	6.4	Read file	c	*Defer to PSESWG or re-scope*
	1003.5	6.1	Read file		
10.2 Execution History	1003.1	6.4	Read file	c	1003.7 or 1003.2 groups (enhance ps) or re-scope
	1003.5	6.1	Read file		
11.1 Fault Information Collection	1003.1	2.4	Errors reported via error return code or errno	a	UNIX International, OSF, POSIX SRASS group, and X3T8
		3.3	Signals generated as result of terminal I/O		
		7.1.1.9	Signals generated as result of terminal I/O		
	1003.4	3.3	Realtime queued signals		
	P1003.4a	3.3	Signals as extended to threads environment		
	P1003.4b	22.3	Interrupt capture, locking, notification		
	1003.5	2.4.4	Error codes and exceptions		
		3.3	Signals generated as result of terminal I/O		
		7.1.1.9	Signala generated as result of terminal I/O		
	P1003.20	3.1	Realtime queued signals		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
11.2 Fault Information Request	1003.1 1003.4 P1003.4a P1003.4b 1003.5 P1003.20	2.4 3.3 7.1.1.9 3.3 3.3 22.3 2.4.4 3.3 7.1.1.9 3.1	Errors reported via error return code or errno Signals generated as result of terminal I/O Signals generated as result of terminal I/O Realtime queued signals Signals as extended to threads environment Interrupt capture, locking, notification Error codes and exceptions Signals generated as result of terminal I/O Signals generated as result of terminal I/O Realtime queued signals	a	UNIX International, OSF, POSIX SRASS group, and X3T8
11.3 Diagnostic Tests Request	P1003.4b	21.1-21.3	Device Control	a	UNIX International, OSF, POSIX SRASS group, and X3T8
11.4 Diagnostic Tests Results	P1003.4b	21.1-21.3	Device control	a	UNIX International, OSF, POSIX SRASS group, and X3T8
11.5 Operational Status	1003.1 1003.4a P1003.4b 1003.5	2.4 3.2.1 6.4 16.1.3 21.1-21.3 2.4.4 3.1.5 6.1	Errors reported via error return code or errno Process termination status Read file Thread termination status Device control Error codes and exceptions Process termination status Read file	a	UNIX International, OSF, POSIX SRASS group, and X3T8
11.6 Fault Detection Thresholds	1003.1 P1003.4a 1003.5	3.3.1.2- 3.3.1.3 3.3.1.2- 3.3.1.3 3.3.2.1	Process signal masking & actions Thread signal masking & actions Process signal masking and actions	a	UNIX International, OSF, POSIX SRASS group, and X3T8
11.7 Fault Isolation	1003.1 P1003.4b 1003.5	2.4 21.1-21.3 2.4.4	Errors reported via error return code or errno Device control Error codes and exceptions	a	UNIX International, OSF, POSIX SRASS group, and X3T8
11.8 Fault Response	1003.1 P1003.4a P1003.4b 1003.5	3.3.1.2- 3.3.1.3 3.3.1.2- 3.3.1.3 21.1-21.3 3.3.2.1	Process signal masking & actions Thread signal masking & actions Device control Process signal masking and actions	a	UNIX International, OSF, POSIX SRASS group, and X3T8

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
11.9 Reconfiguration	1003.1	3.1	Create a process and execute a program	a	UNIX International, OSF, POSIX SRASS group, and X3T8
		3.2.1	Wait for process termination		
		3.2.2	Terminate own process		
		3.3.2	Send a termination signal to a process		
		6.4	Write file		
	P1003.4a	16.1.1-16.1.2	Create a thread executing a function		
		16.1.3	Wait for thread termination		
		16.1.4	Terminate own thread		
		18.2	Thread cancellation		
	P1003.4b	3.1	Spawn a process executing a program		
	1003.5	3.1.2	Create a process and execute a program		
		3.1.3	Terminate own process		
		3.1.5	Wait for process termination		
		3.2.1-3.2.2	Create a process and execute a program		
		3.3.6	Send a termination signal to a process		
		6.1	Write a file		
11.10 Enable/Disable System Component	1003.1	3.1	Create a process and execute a program	a	UNIX International, OSF, POSIX SRASS group, and X3T8
		3.2.1	Wait for process termination		
		3.2.2	Terminate own process		
		3.3.2	Send a termination signal to a process		
	P1003.4a	16.1.1-16.1.2	Create a thread executing a function		
		16.1.3	Wait for thread termination		
		16.1.4	*Terminate own thread*		
		18.2	Thread cancellation		
	P1003.4b	3.1	Spawn a process executing a program		
		21.1-21.3	Device control		
	1003.5	3.1.2	Create a process and execute a program		
		3.1.3	Terminate own process		
		3.1.5	Wait for process termination		
		3.2.1-3.2.2	Create a process and execute a program		
		3.3.6	Send a termination signal to a process		
	P1003.7	6	Device class		
11.11 Performance Monitoring	1003.1	4.5.2	Get process times	a	UNIX International, OSF, POSIX SRASS group, and X3T8
		5.6.2	Get file times		
	P1003.4b	20.1	Process and thread CPU time clocks		
	1003.5	4.2.1	Get process times		
		5.3	Get file times		
11.12 Set Resource Utilization Limits	1003.1	2.8	Numerical limits (variable or invariant)	-	Support P1003.1a balloting. UNIX International, OSF, POSIX SRASS group, and X3T8 Support P1003.4b balloting
	P1003.1a	12	Resource limits		
	1003.4	2.8	Numerical limits (variable or invariant)		
	P1003.4a	2.8	Numerical limits (variable or invariant)		
	P1003.4b	2.8	Numerical limits (variable or invariant)		
		13.2.4	Sporadic server scheduling policy		
		20.1	Process and thread CPU time clocks		
	1003.5	2.4.1.2	System limits		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
11.13 Resource Utilization Limits Violation	1003.1 P1003.1a P1003.4a 1003.5	6.4.2.4 12 16.1.2.4 6.1.2.2	File maximum size violation Resource limits Maximum thread violation File maximum size violation	-	Support P1003.1a balloting. UNIX International, OSF, POSIX SRASS group, and X3T8
11.14 Checkpoint Data Structures	P1003.1a 1003.4 P1003.20	11.2 12.2 12.2	Process checkpoint and restart Memory mapping of file or storage device Memory mapping of file or storage device	-	No Delta
12.1 Virtual Memory Support	P1003.4b	23.1.2	Memory advisory information	a	P1003.7
12.2 Virtual Space Locking	1003.4	12.1	Memory locking and unlocking functions	-	No Delta
12.3 Dynamic Memory Allocation and Deallocation	1003.4	12.2	Memory mapping a special device (allocator)	a	P1003.4d
12.4 Dynamically Protecting Memory	1003.4 P1003.20	12.2.3 12.2.3	Protecting mapped memory Protecting mapped memory	a	P1003.4d
12.5 Shared Memory	1003.4 P1003.20	12.2-12.3 12.2-12.3	Map process addresses to a shared memory object Map process addresses to a shared memory object	b	P1003.7 (code sharing)
12.6 Allocate, Deallocate, Mount and Dismount Services	1003.1 1003.5 P1003.7	6.5 6.1.5 6	Control operations on files (file descriptors) Control operations on files (file descriptors) Device class	a	P1003.7 (mount/dismount)
12.7 Designate Control	None			b	P1003.7
12.8 Release Control	None			b	P1003.7
12.9 Allocate Resource	None			a	1003.1 or 1003.4 group
12.10 Deallocate Resource	None			a	1003.1 or 1003.4 group
12.11 System Resource Requirements Specification	None			b	1003.7 group. New PAR: Resource Management
12.12 System Resource Capacity	None			b	1003.7 group and P1003.4d. New PAR: Resource Management

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
13.1 Process Synchronization	1003.1	3.3	Signals	-	No Delta
		3.4.2	Signals		
		6.1.1	Pipes		
		6.4.1-6.4.2	Read and write pipes, sockets, etc		
	1003.4	3.3	Signals		
		6.7	Asynchronous I/O (Pipes, sockets, etc)		
		11.2	Semaphores		
		12.3	Shared memory		
		15.2	Message passing		
	P1003.4a	3.3	Signals		
		6.4.1-6.4.2	Read and write pipes, sockets, etc.		
		6.7	Asynchronous I/O (pipes, sockets, etc.)		
		11.2-11.4	Semaphores, mutexes, and condition variables		
		15.2	Message passing		
	P1003.4b	11.2.6	Semaphores with timeouts		
		11.3.3	Mutexes with timeouts		
		15.2	Message passing with timeouts		
	1003.5	3.3	Signals		
		6.1.1	Pipes		
		6.1.2	Read and write pipes, sockets, etc.		
	P1003.20	3.1	Signals		
		6.7	Asynchronous I/O (Pipes, sockets, etc)		
		11.2-11.4	Semaphores, mutexes, and condition variables		
		12.3	Shared memory		
		15.2	Message passing		
13.2 Mutual Exclusion	1003.1	5.3.1-5.3.2	Create a lock file (test and set functionality)	-	No Delta
	1003.4	11.2	Semaphores		
		12.3	Shared memory for spin locks		
	P1003.4a	11.2-11.4	Semaphores, mutexes, and condition variables		
	P1003.4b	11.2.6	Semaphores with timeouts		
		11.3.3	Mutexes with timeouts		
		22.3.1	Interrupt locking		
	1003.5	5.2.1	Create a lock directory (test and set functionality)		
	P1003.20	11.2-11.4	Semaphores, mutexes, and condition variables		
		12.3	Shared memory for spin locks		
13.3 Cumulative Execution Time of a Process	1003.1	4.5.2	Get process times	-	No Delta
	P1003.4b	20.1	Process and thread CPU time monitoring		
	1003.5	4.2.1	Get process times		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
13.4 Attach a Process to an Event	1003.1	3.3	Signals generated as the result of events	-	No Delta
		7.1.1.9	Signals generated as result of terminal I/O		
	1003.4	3.3	Realtime queued signals for user caused events		
	P1003.4a	3.3	Signals as extended to thread environment		
	P1003.4b	22.3	Interrupt capture, locking, notification		
	1003.5	3.3	Signals generated as the result of events		
		7.1.1.9	Signals generated as result of terminal I/O		
	P1003.20	3.1	Realtime queued signals for user caused events		
13.5 Services Scheduling Information	1003.4	13.2-13.3	Scheduling policies, parameters, functions	d	1003.21 group
	P1003.4a	13.3	Scheduling policies, parameters, functions		
		13.4-13.5	Thread scheduling and contention scope		
	P1003.20	13.2-13.3	Scheduling policies, parameters, functions		
13.6 Scheduling Delay (See also 9.8 IPC)	1003.1	3.3.7	Wait for a signal (to end delay)	-	No Delta
		3.4.1	Schedule a signal to this process in the future		
		3.4.2	Wait for a signal (to end delay)		
		3.4.3	Wait until time elapses		
		6.4.1	Read a message from pipe (to end delay)		
	1003.2	4.57 (Vol 1)	Sleep utility "sleep"		
	1003.4	3.3.8	Wait for a signal (to end delay)		
		14.2.4	Send a signal to a process in the future		
		14.2.5	Wait until time elapses		
	P1003.4a	3.3.10	Wait for a signal (to end delay)		
	1003.5	6.1.2	Read a message from a pipe (to resume)		
	P1003.20	3.1.3	Wait for a signal (to end delay)		
		14.2.4	Send a signal to a process in the future		
		14.2.5	Wait until time elapses		
	Ada LRM	9.6	Wait until time elapses (delay)		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
13.7 Periodic Scheduling	1003.1	3.3.7	Wait for a signal (beginning of period)	-	No Delta
		3.4.1	Schedule a signal to this process in the future		
		3.4.2	Wait for a signal (beginning of period)		
		3.4.3	Wait until time elapses		
	1003.4	3.3.8	Wait for a signal (beginning of period)		
		14.2.4	Send signal to a process periodically		
		14.2.5	Wait until time elapses		
	P1003.4a	3.3.7	Wait for a signal (beginning of period)		
		3.3.8	Wait for a signal (beginning of period)		
		3.3.10	Wait for a signal (beginning of period)		
		3.4.2	Wait for a signal (beginning of period)		
		3.4.3	Wait until time elapses		
		11.4.4	Periodic execution while awaiting a condition		
	P1003.4b	13.2.4	Sporadic server - periodic replenishment		
		22.3	*Capture and await external H/W timer interrupts*		
	P1003.20	3.1.3	Wait for a signal (beginning of period)		
		11.4.8	Periodic execution while awaiting a condition		
		14.2.4	Send signal to a process periodically		
		14.2.5	Wait until time elapses		
	Ada LRM	9.6	Wait until time elapses (delay)		
13.8 Multiple Scheduling Policies	1003.4	13.2-13.3	SCHED_FIFO, SCHED_RR, SCHED_OTHER process scheduling	-	No Delta
	P1003.4a	13.4	Thread scheduling		
		13.6	Non-inverting mutex scheduling protocols		
	P1003.4b	13.2-13.4	SCHED_SPORADIC (process and thread)		
	P1003.20	13.2-13.3	SCHED_FIFO, SCHED_RR, SCHED_OTHER process scheduling		
13.9 Selection of a Scheduling Policy	1003.4	13.3.3	Set process scheduling policy	-	No Delta
	P1003.4a	13.5.1	Set thread creation scheduling policy attribute		
	P1003.4b	13.3.3	Set process scheduling policy		
		13.5.1	Set thread creation scheduling policy attribute		
	P1003.20	13.2	Set thread scheduling parameters		
		13.3.3	Set process scheduling policy		
13.10 Modification of Scheduling Parameters	1003.4	13.3.1	Set scheduling parameters	-	No Delta
		13.3.3	Set scheduling parameters and policy		
	P1003.4a	13.5.2	Set thread scheduling parameters		
	P1004.4b	13.5.2	Set thread scheduling parameters		
	P1003.20	13.2	*Set thread scheduling parameters*		
		13.3.1	Set scheduling parameters		
		13.3.3	Set scheduling parameters and policy		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
13.11 Precise Scheduling (Jitter Management)	1003.4	13.3.1	Set scheduling parameters	-	No Delta
		13.3.3	Set scheduling parameters and policy		
		14.2.4	High resolution timers		
		14.2.5	High resolution wait until time elapses		
	P1003.4a	13.5.2	Set thread scheduling parameters		
	P1003.4b	13.5.2	Set thread scheduling parameters		
		22.3	Capture and await external H/W timer interrupts		
	P1003.20	13.2	Set thread scheduling parameters		
		13.3.1	Set scheduling parameters		
		13.3.3	Set scheduling parameters and policy		
		14.2.4	High resolution timers		
		14.2.5	High resolution wait until time elapses		
14.1 Image Load	1003.1	3.1.2	Execute a file	a	1003.1 or 1003.7 groups
	P1003.4b	3.1.3	Spawn a process executing a file		
	1003.5	3.1.2	Create a process executing a file		
		3.2.2	Execute a file		
	P1003.7	9	Machine class		
		10	System class		
14.2 System Initialization and Reinitialization	1003.1	3	Process primitives	a	1003.1 or 1003.7 groups
		4	Process environment		
	1003.4	14.2	Clocks and timers		
	1003.5	3	Process primitives		
		4	Process environment		
	P1003.7	9	Machine class		
		10	System class		
		11	Authentication class		
		12	Network class		
		13	Backup class		
	P1003.8	Unknown	Unknown		
	P1003.20	14.2	Clocks and timers		
14.3 Shutdown	1003.1	3.2.2	Terminate a process (self)	a	1003.1 or 1003.7 groups
		3.3.2	Send a termination signal to a process		
	P1003.4a	16.1.4	Terminate a thread (self)		
		18.2.1	Cancel execution of a thread		
	1003.5	3.1.3	Terminate a process (self)		
		3.3.6	Send a termination signal to a process		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
15.1 Read Selected Clock	1003.1	4.5.1	Get system time	-	No Delta
		4.5.2	Get process CPU times		
	1003.4	14.2.1	Get time and resolution for a specific clock		
	P1003.4b	20.1.3	Get process CPU time		
		20.1.6	Get thread CPU time		
		21.2	Get time from clock on external device (devctl)		
	1003.2	4.15 (Vol 1)	Display date and time		
	1003.5	4.2.1	Get process CPU times		
		4.4.1-4.4.2	Get system time		
	P1003.7	9 or 10	Machine class or system class		
	P1003.20	14.2.1	Get time and resolution for a specific clock		
15.2 Set Selected Clock	1003.4	14.2.1	Set a specified clock	-	No Delta
	P1003.4b	20.1.3	Set a process CPU time clock (questionable)		
		20.1.6	Set a thread CPU time clock (questionable)		
		21.2	Set a clock on external device (devctl)		
	P1003.7	9 or 10	Machine class or system class		
	P1003.20	14.2.1	Set a specified clock		
15.3 Synchronization of Selected Clocks	1003.4	14.2.1	Get time, resolution, and set a specific clock	-	No Delta
	P1003.4b	21.2	Get time and set clock on external device		
	P1003.20	14.2.1	Get time, resolution, and set a specific clock		
15.4 Select a Primary Reference Clock	1003.4	14.2.1	Support for multiple clocks	a	1003.21 group new PAR
	P1003.4b	14.2.1	Support for process/thread CPU time clocks		
	P1003.20	14.2.1	Support for multiple clocks		
15.5 Locate the Primary Reference Clock	1003.4	14.2.1	Support for multiple clocks	d	1003.21 group plus requirement rewording
	P1003.4b	14.2.1	Support for process/thread CPU time clocks		
	P1003.20	14.2.1	Support for multiple clocks		
15.6 Timer Services	1003.1	3.3.4	Establish action for alarm or timer signal	-	No Delta
		3.3.7	Wait for alarm or timer signal		
		3.4.1	Schedule an alarm signal to this process		
		3.4.2	Wait for alarm or timer signal		
	1003.4	3.3.8	Wait for alarm or timer signal		
		14.2.2-14.2.3	Create and delete timers		
		14.2.4	Set and start a periodic or one-shot timer		
	P1003.4a	3.3.7	Wait for alarm or timer signal		
		3.3.8	Wait for alarm or timer signal		
		3.3.10	Wait for alarm or timer signal		
		3.4.2	Wait for alarm or timer signal		
	P1003.4b	22.3	Capture and await external H/W timer interrupts		
	P1003.20	3.1.3	Wait for alarm or timer signal		
		14.2.2-14.2.3	Create and delete timers		
		14.2.4	Set and start a periodic or one-shot timer		
	Ada LRM	13.5.1	Establish action for timer signal		
15.7 Precision Clock	1003.4	14.1.1	Timespec structure	-	No Delta
		14.2.1	Get clock resolution		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
16.1 Create task (Ada)	P1003.4a	13.5.1-13.5.2	Setting thread scheduling attributes	-	No Delta
		16.1.1-16.1.2	Thread creation with attributes		
	P1003.20	13.2	Setting thread scheduling attributes		
16.2 Abort task (Ada)	P1003.4a	16.1.3	Wait for thread termination	a	P1003.4d
		18.2.1-18.2.3	Thread cancellation		
16.3 Suspend Task (Ada)	1003.4	11.2.6	Wait for a locked semaphore (until resumed)	-	No Delta
	P1003.4a	11.4.4	Wait on a condition (to resume)		
	P1003.20	11.2.6	Wait for a locked semaphore (until resumed)		
		11.4.8	Wait on a condition (to resume)		
16.4 Resume Task (Ada)	1003.4	11.2.7	Unlock a semaphore (waited for)	-	No Delta
	P1003.4a	11.4.3	Broadcast or signal a condition (waited for)		
	P1003.20	11.2.7	Unlock a semaphore (waited for)		
		11.4.7	Broadcast or signal a condition (waited for)		
16.5 Terminate Task (Ada)	P1003.4a	16.1.3	Wait for thread termination	-	No Delta
		16.1.4	Terminate a thread (self)		
		18.2.1-18.2.3	Thread cancellation		
16.6 Restart Task	None			d	Investigate checkpoint/restart of threads in 1003.7 group
16.7 Task Entry Calls (Ada)	P1003.4a	Unknown	Pending decision on mapping Ada tasks to Pthreads	-	No Delta
	P1003.20	Unknown	Pending decision on mapping Ada tasks to Pthreads		
16.8 Task Call Accepting/Selecting	P1003.4a	Unknown	Pending decision on mapping Ada tasks to Pthreads	-	No Delta
	P1003.20	Unknown	Pending decision on mapping Ada tasks to Pthreads		
16.9 Access Task Characteristics (Ada)	1003.4	14.2.2-14.2.3	Create and delete timers (CPU budget timers)	-	No Delta
		14.2.4	Set and start a periodic or one-shot timer		
	P1003.4a	13.5.2	Set and get thread scheduling parameters		
		16.1.5	Get my thread ID		
		18.2.2	Set thread cancelability state and type		
	P1003.4b	13.5.2	Set thread scheduling parameters		
		20.1.3`	Access process CPU clock		
		20.1.5	Access CPU clock attributes		
		20.1.6	Access thread CPU clock		
	P1003.20	14.2.2-14.2.3	Create and delete timers (CPU budget timers)		
		14.2.4	Set and start a periodic or one-shot timer		
		13.2	Set and get thread scheduling parameters		
16.10 Monitor Task's Execution Status (Ada)	P1003.4a	16.1.3	Return thread termination status	a	1003.7 or 1003.2 groups
	P1003.4b	20.1	Thread CPU time monitoring		
16.11 Access to a Precise Real-Time Clock (Ada)	1003.4	14.2.1	Get/Set time, get resolution for specific clock	-	No Delta
	P1003.4b	21.2	Get/Set time from clock on external device		
	P1003.20	14.2.1	Get/Set time, get resolution for specific clock		
16.12 Access to a TOD clock (Ada)	1003.1	4.5.1	Get system time	-	No Delta
	1003.4	14.2.1	Get/set time, get resolution for specific clock		
	P1003.4b	21.2	Get/Set time from clock on external device		
	1003.5	4.4.1-4.4.2	Get system time		
	P1003.20	14.2.1	Get/Set time, get resolution for specific clock		

DELTA SUMMARY AND CROSS REFERENCES (Cont'd)

OSSWG Requirement	Reference	Paragraph	Capabilities Fulfilling Requirement	Rating	Recommendation
16.13 Dynamic Task Priorities (Ada)	P1003.4a	13.5.2	Set thread scheduling parameters	-	No Delta
	P1003.4b	13.5.2	Set thread scheduling parameters		
	P1003.20	13.2	Set thread scheduling parameters		
16.14 Scheduling Policy Selection (Ada)	1003.4	13.3.3	Set process scheduling policy	-	Address the Ada language delta
	P1003.4a	13.5.1	Set thread creation scheduling policy attribute		
	P1003.4b	13.3.3	Set process scheduling policy		
		13.5.1	Set thread creation scheduling policy attribute		
	P1003.20	13.2	Set thread creation scheduling policy attribute		
		13.3.3	Set process scheduling policy		
16.15 Memory Allocation and Deallocation	1003.4	12.2	Memory mapping a special device (allocator)	a	P1003.4d
	P1003.20	12.2	Memory mapping a special device (allocator)		
16.16 Interrupt Binding (Ada)	P1003.4b	22.3	Interrupt capture, locking, notification	-	No Delta
16.17 Enable/Disable Interrupts (Ada)	P1003.4b	22.3	Interrupt locking/unlocking	-	Address the Ada language delta
16.18 Mask/unmask Interrupts (Ada)	1003.1	3.3.5	Examine and change blocked signals	c	Inappropriate for standardization
	1003.5	3.3.7	Examine and change blocked signals		
16.19 Raise Exception (Ada)	1003.1	3.3	Signals	-	No Delta
	1003.4	3.3	Signals		
	1003.4a	3.3	Signals		
	1003.5	2.4.4	Error codes and exceptions		
		3.3	Signals		
	P1003.20	3.1	Signals		
16.20 I/O Support (Ada)	1003.1	5	POSIX file support	-	No Delta. See also all requirements in service classes 6 and 7
		6	POSIX I/O primitives		
	1003.4	5	POSIX file support		
		6	POSIX I/O primitives		
	P1003.4a	5	POSIX file support		
		6	POSIX I/O primitives		
	P1003.4b	21.2	Low level device control and status		
	1003.5	5	Ada POSIX file support		
		6	Ada POSIX I/O primitives		
	P1003.20	5	POSIX file support		
		6	POSIX I/O primitives		

Addressee	No. of Copies
Commanding Officer ••	2

NAWCWPNS
1 Adminstration Circle
China Lake, CA 93555
(1 for Code C21C; Lee Lucas)
(1 for Code C026; John Zenor)

Addressee	No. of Copies
Commanding Officer NAWC-AD Pax River Kevin Dodson Computer Sciences Directorate Pax River, MD 20670	1
Dr. Karen Gordon Institute for Defense Analyses 905 S. Lane Ct. Brentwood, TN 37027	1
Joe Gwinn Raytheon Co. MS 5-2-505 528 Boston Post Rd. Sudbury, MA 01776	1
Mary Hermann Raytheon MS P3SU10 50 Apple Hill Dr. Tewksbury, MA 01876	1
Commanding Officer NSWC-White Oak 10901 New Hampshire Ave. Silver Spring, MD 20903 (5 for Code U33; Steve Howell) (1 for Code B44; Helmut Roth, Rm 2-151)	6
Commanding Officer NAWC, AD Indianapolis, 6000 E 21st. Street Indianapolis, IN 46219 (1 for Code DP301N/MS-31; Diane Kohalmi) (1 for Code DP304N/MS-26; Greg Vantreese)	2
J.F. Leathrum Clemson University ECE Department Riggs Hall, Box 340915 Clemson, SC 29673-0915	1
Craig Meyers Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15123	1

Addressee	No. of Copies
V. Raj Avula PEO (SD) D2 2531 Jefferson Davis Hwy. Arlington, VA 22242-5170	1
NUWC, Newport Code 2221, Bldg. 1171/3 Newport, RI 02841-5047 (1 for John Brennan) (1 for Greg Bussiere) (1 for Dan Juttlestad) (10 for Jim Oblinger) (1 for Bruce Stevens)	14
Unisys Corporation P.O. Box 64525 St. Paul, MN 55164-0525 (1 for Steve Case) (1 for Dale Georgen; MS U1M30) (1 for Kari Kruempel; MS U1M30) (1 for Dr. Del Swanson; MS U1M30)	4
Elizabeth Czul Booz, Allen & Hamilton Suite 711 2231 Crystal Drive Arlington, VA 22202	1
Jim Johnston Booz, Allen & Hamilton 8283 Greensboro Drive, 8th Floor McLean, VA 22102-3838	1
NCCOSC RTD&E Div. Code 4122 49180 Transmitter Rd., Rm. 2 San Diego, CA 92152-7341 (1 for Shane Deichman) (10 for Donna Fisher)	11
NCCOSC RTD&E Div. Gil Myers, Code 4103 53140 Gatchell Rd., Rm 355 San Diego, CA 92152-7440	1
NCCOSC RTD&E Div. Dr. Mike Shapiro, Code 4123 271 Catalina San Diego, CA 92152	1

Addressee	No. of Copies
Air Combat Command HQ ACC/SCP Attn: Capt. Steven Muhssm Langley AFB, VA 32663-6343	1
Director, Defense Logistics Agency Attn: ACQOA Cameron Station Alexandria, VA 22304-6100	1
Director, JIEO Center for Standards Attn: TBBD (Richard McLane) Fort Monmouth, NJ 07703-5613	1
Director, JIEO Center for Standards Attn: TBE (Capt. Petersen, USN) 11440 Isaac Newton Square, North Reston VA 22090-5006	1
Director, National Security Agency Attn: Code K642 (Paul Minnigh) Fort George G. Meade, MD 20755-6000	1
Director, Defense Mapping Agency (DepSo) Attn: PRS, POC: Billy J. Love Mail Stop A-13 8613 Lee Highway Fairfax, VA 22031-2137	1
Director, Defense Intelligence Agency Attn: James C. Moore, DAP-DMO P.O. Box 46563 Washington, DC 20050-6563	1

<u>OSSWG Members</u>

Reed Adams VITRO 5001 N. SR 37-BUS Bloomington, IN 47404-1626	1
Dr. Charles Arnold Arnold Associates P.O. Box 1990 Marina, CA 93933	1

Addressee	No. of Copies
U.S. Army Information Systems Engineering Command Attn: ASQB-OSI-S Fort Huachuca, AZ 85613-5300	1
Director, U.S. Army Industrial Engineering Activity Attn: AMXIB-PA (Tom Schneider) Rock Island, IL 61299-7260	1
Simulation, Training and Instrumentation Command Attn: SMSTI-SI (Chris Catotti) 12350 Research Parkway Orlando, FL 32826-3276	1
Commandant, U.S. Coast Guard Headquarters Attn: G-ELM-4 Washington, DC 20593-0001	1
HQ USAF/SCXS Attn: Standards (Fred Virtue) 1250 AF Pentagon Washington, DC 20330-1250	1
HQ AFMC/ENSR Attn: Major George Newberry 4375 Chidlaw Road, Suite 6 Wright-Patterson AFB, OH 45433-5006	1
ASC/SMA Attn: Dave Bond Building 46 1895 5th Street Wright-Patterson AFB, OH 45433-7200	1
Computer Architecture and Standards Division 615 SMSQ/CITA Attn: Denzel Henderson 4225 Logistics Ave., Suite 15 Wright-Patterson AFB, OH 45433-5754	1
HQ Air Force C4 Agency HQ AFCUA/TNABF Attn: Rex McKinnon 607 Pierce Street, Room 303 Scott AFB, IL 62225-5421	1
HQ Air Force Communications Command SSC/XPT Gunter AFB, AL 36114-6343	1

DISTRIBUTION LIST
Report No. NADCADWAR-94109-70

Addressee	No. of Copies
Space and Naval Warfare Systems Command 2451 Crystal Drive Arlington, VA 22245 (5 for SPAWAR 331-2) (1 for SPAWAR 10-224)	6
Naval Air Warfare Center Aircraft Division Warminster P.O. Box 5152 Warminster, PA 18974-0591 (2 for Code 0471) (10 for Code 7032; F. Prindle) (1 for Code 7033; D. Hsu)	13
Defense Technical Information Center ATTN: DTIC-FDAB Cameron Station BG5 Alexandria, VA 22304-6145	2
Center for Naval Analysis 4401 Fort Avenue P.O. Box 16268 Alexandria, VA 22302-2068	1
SD-1	
Weapons Systems Improvements and Analysis Group Attn: George Desiderio 5109 Leesburg Pike Suite 306 Falls Church, VA 22041-3466	1
CALS Evaluation and Integration Office Department of Defense Attn: Ms. Marianne Pietras Room 3D833, Pentagon Washington, DC 20301-8000	1
Director of Defense Information OSAD (C3I) Information Technology Directorate Attn: Bert Newlin Crystal Gateway 2, Suite 910 1225 Jefferson Davis Hwy. Arlington, VA 22202	1