doc. nr.	ISO/IEC JTC 1/SGFS N 815	
date	1993-02-18	total pages
item nr.		supersedes document

Secretariat: Nederlands Normalisatie-instituut (NNI)
Kalfjeslaan 2 P.O. box 5059
2600 GB Delft
Netherlands
telephone: + 31 15 690390
telefax: + 31 15 690190
telex: 38144 nni nl
telegrams: Normalisatie Delft

ISO/IEC JTC 1/SGFS
Title: ISO/IEC JTC 1 Special Group on Functional Standardization

Secretariat: NNI (Netherlands)

Title : JTC1 N2319:
U.S. Contribution on the Need for JTC1 to Establish Specific Policies and Precedures for the Standardization of Application Program Interface (API) Specifications

Source : U.S. National Body

Status : For information

Note :

ISO/IEC JTC 1 N 2319
Date: 1993-01-22

ISO/IEC JTC 1
INFORMATION TECHNOLOGY
Secretariat: USA (ANSI)

TITLE: U.S. Contribution on the Need for JTC 1 to Establish Specific Policies and Procedures for the Standardization of Application Program Interface (API) Specifications

SOURCE: U.S. National Body

PROJECT: --

STATUS: U. S. Contribution to the JTC 1 Plenary Meeting in Berlin

REQUESTED ACTION: For information and review at the JTC 1 Plenary Meeting in Berlin, 23-26 March 1993

DISTRIBUTION: P and L Members

Address reply to:
Secretariat ISO/IEC JTC 1—American National Standards Institute,11 West 42nd Street, New York, NY 10036
Tel: 212 642-4934, 212 642-4884; TX: 42 42 96 ANSI UI; FAX: 212 398-0023

U.S. CONTRIBUTION ON THE NEED FOR JTC1 TO ESTABLISH SPECIFIC POLICIES AND PROCEDURES FOR THE STANDARDIZATION OF APPLICATION PROGRAM INTERFACE (API) SPECIFICATIONS

DECEMBER 4, 1992

TABLE OF CONTENTS

0. OVERVIEW

Standardization of Application Program Interface (API) specifications by ISO/IEC JTC1 is important for implementation of many Information Technology products and services which utilize JTC1 standards. Indeed, successful standardization of API specifications has already been underway for some time within JTC1. ISO/IEC draft Technical Report 10182, Guidelines for Language Bindings, offers guidelines on preparation of API specifications and draws from the successful experiences within JTC1 in the fields of computer graphics and database management. The U.S. believes that a significant increase in the number of projects for standardization of API specifications can be expected within JTC1 in the near future and hopes that the adoption of the recommendations contained in this document will improve the standardization of API specifications.

This document addresses important concepts, requirements and procedures related to API standardization.

In order to discuss this area of standardization, the U.S. believes it is important to distinguish between the terms "API" and "API specification." Section 1 of this contribution establishes operational definitions for these two terms.

There are three main sections to this contribution. These sections describe important API concepts, methodologies for standardization of API specifications, and a management approach to standardization of API specifications. In order to assist in the review of this contribution, the recommendations found in the three main sections are summarized in Appendix D.

If the recommendations in this contribution are accepted, they would require modifications to the JTC1 Directives and the administrative processing of projects. These modifications would best be adopted as a supplement to the JTC1 Directives in order to expedite their implementation. If JTC1 agrees, the U.S. would be pleased to recommend specific text to SWG-P for its consideration.

1. CHARACTERISTICS AND DEFINITION

1.1 Application Program Interface (API) Related Concepts

An interface is a boundary between two entities that defines a relationship between them. For an Application Program Interface (API), the boundary is between an application software entity (the service user) and an application platform (the service provider). This application platform is a set of resources on which the application software will run, providing all of the services required by the application software. In this context, a service is a capability of a service-providing-entity which is made available to a service-using-entity at the boundary between those entities.

The TSG-1 model [ISO/IEC JTC1 N1335, May 1, 1991] "TSG-1: Standards Necessary to Define Interfaces for Application Portability (IAP) - Final Report", describes two kinds of interfaces that are important for application portability:

a) The Application Program Interface (API) is the "internal" interface between application software and the application software platform; and

b) The Platform External Interface (PEI) is the "external" interface between application software platform and the external world.

This document addresses only API specifications. Note that the relationships depicted in Figure 1 could be represented in a number of ways. Clear identification of the key interfaces and entities is the objective of this discussion rather than a selection of any particular diagram.

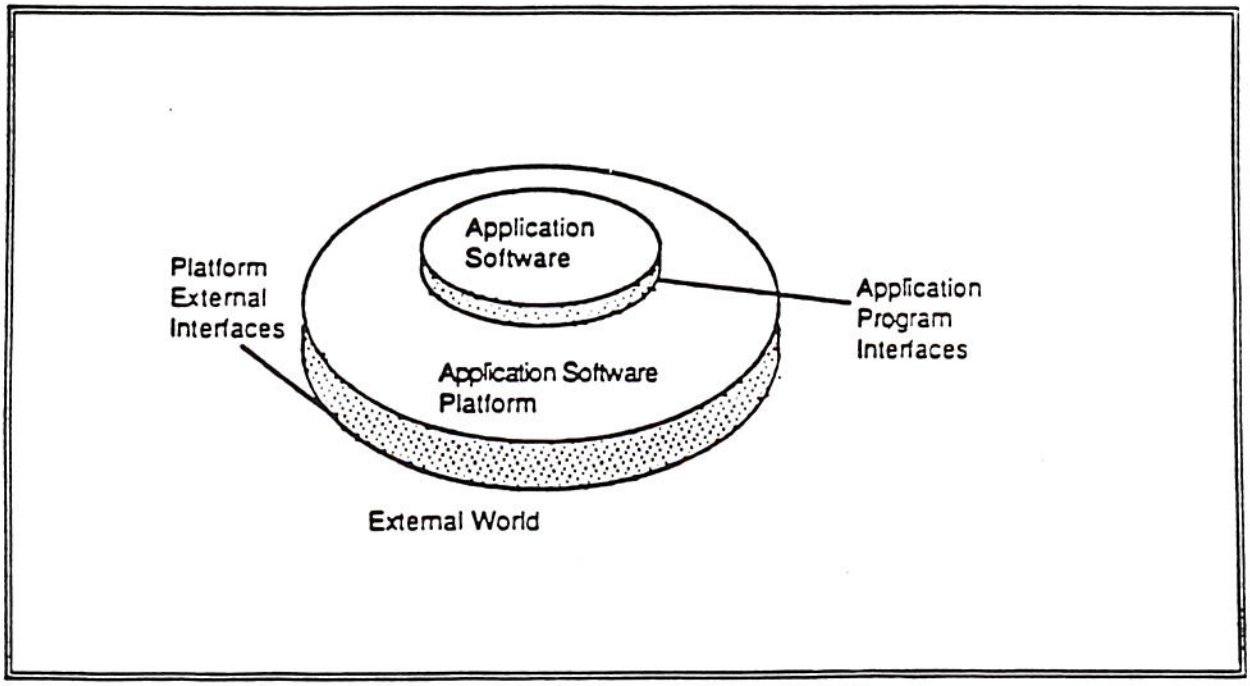

Figure 1: API Relationships

Within the context of this document, there may be multiple API relationships within any system. The distinction between an application and its supporting platform is relative, rather than absolute, with respect to the specific application under consideration.

An API is a boundary across which application software uses facilities of programming languages to invoke services. These facilities may include procedures or operations, shared data objects and resolution of identifiers. A wide range of services may be required at an API to support applications. Different methods may be appropriate for documenting API specifications for different types of services.

The information flow across the API boundary is defined by the syntax and semantics of a particular programming language, such that the user of that language may access the services provided by the application platform on the other side of the boundary. This implies the specification of a mapping of the functions being made available by the application platform into the syntax and semantics of the programming language.

An API specification documents a service and/or service access method that is available at an interface between the application and an application platform.

An API specification may take the form of one of the following:

a) Programming language specification, which is a description of a language defined within the program of work of SC22, such as Fortran, Ada and C.

b) Language independent API specification, which is a description of a set of functionality in terms of semantics (in an abstract syntax) and abstract data types that can be bound to multiple programming languages.

c) Language specific API specification, which is a description of a set of functionality in terms of the syntax and data types of some programming language.

Language-independent API specifications are useful in defining specifications for invoking services at the API. The language independent specification serves primarily as the reference used to assure consistency across different language bindings. However, one or more language bindings to programming languages such as COBOL or C must also exist. Language specific API specifications are used by programmers, writing in a particular programming language, to invoke a service provided by the application platform. They may be used by a program to invoke a supporting service offered by another application software entity.

1.2 Requirements for Standard API Specification

The needs for standardization of API specifications include:

a) For integration of information technology standards

API standards provide a linkage between standards which specify services and standards which specify programming languages. Thus, API standards play a vital role in integrating Information Technology standards into a coherent whole. This integration is important both for effectiveness and manageability of the standards development process, and for effectiveness of the resulting standards.

b) For application portability

API standards have portability as a key objective. To accomplish this, individual API specifications and combined sets of API specifications (Open System Environment profiles), must be sufficiently complete to enhance predictable portability. The porting of an application that uses standard API specifications to another system that provides the same standard API specifications is considerably simpler than ports involving different systems not based on standard API specifications.

c) For application interoperability

By facilitating application use of interoperable services, API standards help support interoperability between applications.

d) For specifying a suitable platform to provide a set of services

Computer users can select and match application software and application platforms from potentially different suppliers to assemble an environment to address specific business needs.

e) For software reuse

Standard API specifications also promote reuse of software modules.

1.3 Level of Abstraction

The concept of "Level of Abstraction" is complex, with several (possibly non-conflicting) usages. Usage of "Level of Abstraction" implies variation in the amount of functionality offered to the calling program by each invocation.

The same service may be provided by multiple API specifications which differ in level of abstraction. For example, a less abstract API specification for X.400 electronic mail services may provide the application programmer substantial control over details of its interaction with the mail servers. On the other hand, a more abstract API specification may provide a simple, single subroutine call for sending a file as a mail message to a mailbox.

Under this usage, a more abstract API specification is easier to use than a less abstract specification provided that the conventions adopted in implementing the service are appropriate to the application. A less abstract API specification is used where there are application specific requirements relating to details of the interaction or the implementation.

Another usage of "Level of Abstraction" reflects the degree to which the implementation method is visible/invisible to the users of the API specification calls.

An API specification may reflect a pure abstraction driven only by the service requirements (e.g. a protocol independent network API specification), or it may reflect details of the implementation. These details may be associated with one of several alternative methods for service satisfaction (e.g. OSI, TCP/IP, or ISDN communication service API specification). The details could also reflect aspects of alternative platforms' implementations. In this context, use of a more abstract API specification yields greater portability and implementation independence, while a less abstract API specification may provide more control and/or improved performance.

The level of abstraction of API specification varies with the programming language and the abstractions inherent to the specific service. Therefore, a "uniform" level of abstraction across the set all API specifications is not appropriate.

2. METHODS AND COMPONENTS FOR JTC1 API WORK

2.1 Introduction

A standard API specification specifies a mapping between a programming language and the features of a particular service, and thereby provides access to that service from applications written in a particular programming language. Such a mapping is said to create a binding between the service and the programming language.

2.2 Relation to Other Standards

A standard API specification may be part of the standard that specifies the associated programming language, may be part of the standard that specifies the associated service, or may be a separate standard that refers to other standards that define the associated programming language and service. Thus, programming language standards can be considered as one kind of standard API specification.

The following policies are recommended:

1. Standard API specifications shall identify the standards that specify the programming language and the service associated with it, if these are not specified by the standard API specification itself.

2. Standard API specifications shall be consistent with, and shall avoid duplication of, requirements specified by the associated service and programming language standards.

3. Where it can be expected that implementations will support bindings to a service for multiple programming languages, any requirements on interoperability between these bindings should be specified, including requirements on exchange of data values.

4. Where it can be expected that implementations will support bindings to multiple services for a single programming language, any requirements on compatibility between these bindings should be specified, including requirements on coordination of identifier name spaces.

2.3 Language-Independent API Specifications and Language Bindings

Standard API specifications can specify a direct mapping between a programming language and a service, or an indirect mapping that makes use of an intermediate, abstract interface model and syntax that is separately mapped to the programming language and to the service. Where an indirect mapping is used and the same abstract interface is mapped to multiple programming languages, the specification of the mapping from the service to the abstract interface model and syntax is called a language-independent API specification. A specification of a mapping to a programming language, whether directly from a service or from a language-independent API for a service, is called a language binding for that service.

Where there are multiple language bindings to a service, some language bindings may depend on a language-independent API specification, while others map directly to the service, and different groups of language bindings may depend on separate language-independent API specification, for example where the bindings for different programming languages have incompatible requirements. [See Figure 2.]

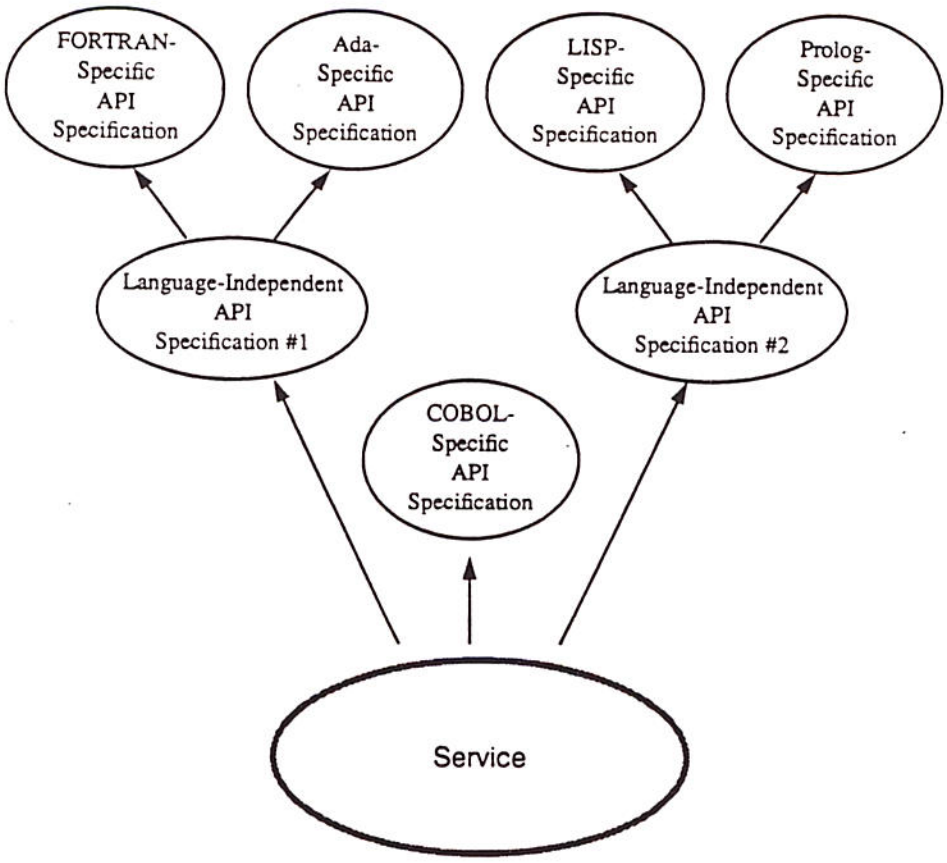

Figure 2: Role of Language-Independent API Specifications

The following policies are recommended:

5. Where a standardization project for an API specification includes multiple language bindings with common interface characteristics, the use of language-independent API specifications should be strongly encouraged.

6. Where a standardization project for an API specification includes a language-independent API specification, the language-independent API specification shall be progressed together with at least one language binding that depends on the language-independent API specification.

7. The development and use of common, standardized methods for the specification of language-independent API specification should be encouraged.

The Common Language-Independent Datatypes [SC22 N970, CD 11404], Common Language-Independent Procedure Calling Mechanisms [SC22 N1082, WD], and the OSI Interface Definition Notation [SC21 N7348, CD 11578] (when remoteness is involved) are intended to define components of an abstract model and syntax that can be used to specify language-independent API specifications. A primary goal of this work is to facilitate the generation of language bindings, when the model and syntax are separately mapped to the service and to the programming language. This approach will reduce the need for separate language binding standards. However, this methodology is not likely to yield results compatible with existing practice.

2.4 Conformance and Testability

Clear definitions of conformance and testability are essential for standard API specifications. Not all required functions can be effectively tested. However, where possible, test methods should be readily derivable from the standard. [See ISO/IEC 9646:1991, and SC22 N1180, which is the basis for an SC22 CD in Q4 1992.]

The following policies are recommended:

8. Standard API specifications shall specify the mapping between conformance levels defined by the API standard and conformance levels defined by the standards defining the associated programming language and service.

9. The "conformance clauses" and conformance requirements specified in standard API specifications shall distinguish between the requirements on a platform's conforming service implementations and those on conforming applications.

10. API conformance requirements should include sufficient level of specificity that verification test methods can be readily derived.

11. The use of API specification methods that support the use of automated test procedures should be encouraged.

2.5 Models

All API specifications, in mapping between a programming language and a service, must take into account the underlying semantic models of the programming language and the service, whether these are explicit or implicit.

The following policies are recommended:

12. The specification of explicit semantic models should be encouraged in the development of standards for programming languages and services, in order to facilitate the development of API specifications which bind them together.

13. The mapping defined by an API specification may harmonize or, instead, separate the underlying semantic models of the associated programming language and service. The approach chosen should be determined in each case according to the characteristics of these semantic models and the needs of the users of the API specification.

14. Where a JTC1 standard exists for a model or framework that addresses the scope of proposed work, the relationship of that work to the model shall be documented.

2.6 Formal Methods

Formal methods can improve the correctness and clarity of a standard, encourage consistency between standards, and simplify testing, and in the case of API specifications, may provide for unambiguous derivation of bindings and test methods. However, for some areas of work formal methods may not exist, or be complete enough to provide a sufficient description. Where work is based on existing experience, formal methods may introduce substantial delays into development of the standard. And, in some cases, formal methods may reduce the ease-of-use of a standard (note that standard API specifications are used by both platform implementors and application implementors.)

The following policy is recommended:

15. In determining whether the use of a formal specification method is appropriate, the timeliness and useability of the resulting standard should be considered, as well as the availability of National Body experts.

3. NEED FOR AN IMPLEMENTATION PLAN

3.1 Introduction

There are a large number of existing and future projects involving the standardization of API specifications. Consequently, there is a need to establish criteria and specific JTC1 policies and procedures for such standardization which will allow ongoing work to continue while migrating to a comprehensive approach. The development by JTC1 of such an implementation plan should include review of the proposed plan by both JTC1 NBs and JTC1 Subgroups.

3.2 Placement within JTC1 for Standardization of API Specifications

Standardization of API specifications within JTC1 has been successfully progressed using various approaches to placement of the work. It has been accomplished by groups standardizing programming languages (for example, the binding of GKS to Basic) and groups specifying the services (for example, the binding of SQL to Ada).

There are three types of expertise that need to be involved in the development of API specifications. The developers of the service standards must be involved to provide detailed knowledge on the use of service standards. The users of the base standards and the API must be involved to facilitate determining the appropriate levels of abstraction and "common usage". Language experts need to be involved to ensure that the API is properly integrated into the various languages. One of the more important aspects of the language dependent API specifications is that they fit well into the style and model of the language in which they are to be used without conflicting with existing syntax and semantics. Those participating in the development of an API specification would not be expected to limit their participation to a single type of expertise. However, these categories do serve to characterize the kinds of knowledge that needs to be involved.

Level of Abstraction and Language-Independence are two characteristics of API specifications that could be used to help determine the placement of work on API specifications. Much like the types of expertise just described, these are not an absolute characterization, but more an indication of degrees or shadings. For API specifications that were very language independent and were at a lower level of abstraction, the work may rely more heavily on base standards participation. By moving to higher levels of abstraction, more user participation and a rise in the need for language expertise is expected. By moving from more language independence to more language dependence, greater participation by language experts and less participation from base standards is expected.

The following policy is recommended:

16. To expedite placement of future work, an NP or fast-track submission which includes an API component must be accompanied by a statement that addresses the following questions:

    a) Which SC is responsible for the underlying service?

    b) Which SC is responsible for the programming language(s)?

    c) Will the API specification require extension(s) to an existing programming language or service?

d) What is the kind of expertise required in the development of the API specification?

e) What resources of the SC are available to perform the new API specification work?

f) What is the relationship of the API specification work to other work in the SC?

g) Is the appropriate expertise available for review and consideration of the draft API specifications, especially during the CD ballot stage?

3.3 Coordination

Given the nature of standardization of API specifications, there are clear needs to ensure that NPs for the work include detailed information on:

a) Related work (base standards and ISPs) and

b) necessary liaisons.

JTC1 has recently reorganized to enhance its abilities to coordinate its program of work (e.g., additional coordination questions are being added to the NP form). For standardization of API specifications it would be useful to address specific questions (as discussed in Section 3.2). Similarly, requesters of fast-track processing for API standards should be asked to provide information analogous to that provided for NP processing.

Standardization of API specifications are dependent upon related standards which apply to one or both sides of the interface involved. Therefore, it is important that during the development of standards for API specifications:

a) Related work is progressed and

b) liaisons are active

in order to ensure that technically sound and complete standards are developed in a timely fashion.

The following policy is recommended:

17. Standardization of API specifications should require specific explicit review by specifically identified liaisons at specific stages of development (e.g., CD, at time of registration, should be sent to these identified liaison SCs for required review and comment to the developing SC.)

3.4 International Standardized Profiles (ISPs)

The SGFS has recently had its area of work expanded to include ISPs for Open System Environments (OSE). It is anticipated that this process will identify the need to further Standardization of API specifications.

3.5 Migration

It is obvious that there will be an increasing workload in JTC1 on Standardization of API specifications.

A JTC1 implementation plan for new policies and procedures for Standardization of API specifications should include:

a) Applicability to new JTC1 projects and JTC1 projects which have not yet reached the stage of CD registration,

b) Applicability to all other projects which have not reached IS stage by a future deadline, and

c) a target date for implementation.

APPENDIX A. REFERENCES

ISO 9646:1991 - Information technology - Open system interconnection - Conformance testing methodology and framework

ISO/IEC JTC1 Directives

ISO/IEC DTR 10182, "Guidelines for Language Bindings," Draft technical report, document SC22 N1098.

ISO/IEC JTC1 N1335, May 1, 1991: "TSG-1: Standards Necessary to Define Interfaces for Application Portability (IAP) - Final Report."

SC22 N970, CD 11404: Information technology - Programming languages - Common language-independent datatypes

SC22 N1082, WD: Information technology - Programming languages - Common language-independent procedure calling mechanisms

SC21N 7348, CD11578: Information technology - Open Systems Interconnection -Interface Definition Notation

SC22 N1180, WD: Information technology - Programming languages - Test methods for testing conformance to POSIX

APPENDIX B. API PROJECTS

The following is a list of SC18, SC21 and SC24 projects which are candidates for requiring API specification be developed for them. This is not an official list and may have either errors of omission (a project should be on the list which isn't) or inclusion (a particular project shouldn't be on the list which is).

The purpose of this list to illustrate the magnitude of the work which will be required to provide the API specifications required for current standards projects from just SC18, SC21 and SC24 (there are many other committees developing standards which will also require API specifications).

Project No.	Title
SC18 Projects	
18.10	Abstract Interface for Manipulation of ODA Documents
18.11	Message Handling System (MHS)
18.15.01	Standard Generalized Markup Language (SGML)
18.15.06.02	Standard Page description Language (SPDL)
18.21.02	Graphical Symbols Used in Screens
18.29	Keyboard Maps
18.31	Document Filing and Retrieval (DFR)
18.32	Document Printing Application (DPA)
18.33.01	Font Services, Part 1: Abstract Service Definition
18.33.02	Font Services, part 2: Protocol Specification
18.35	Support for Electronic Data Interchange (EDI) in MHS
18.37	Referenced Data Transfer (RDT)
18.41	Dialog Interaction Techniques
18.46	User Interface to Voice Messaging (and related applications)
SC21 Projects	
1.21.03.02	SQL
1.21.06	IRDS
1.21.11	Virtual Terminal
1.21.12	FTAM
1.21.13	FTM
1.21.17.3	ASN.1
1.21.22.1	ACSE
1.21.28.3	CMIS
1.21.28.5.1	Object Management Function
1.21.28.5.2	State Management Function
1.21.28.5.5.4	General Relationship Management Function
1.21.28.5.5.5	Change Over Function
1.21.28.6.1	Alarm Reporting Function
1.21.28.6.2	Event Report Management Function
1.21.28.6.3	Log Control Function
1.21.28..6.4	Test Management Function
1.21.28.7.1	Security Alarm Reporting Function
1.21.28.7.2	Security Audit Trail Function
1.21.28.8.1	Accounting Meter Function
1.21.28.9.1	Workload Monitoring Function
1.21.28.9.2	Summarization Function
1.21.28.9.3	Response Time Monitoring Function
1.21.28.9.4	Scheduling Function

1.21.28.10	Software Management Function
1.21.28.11	Time Management Function
1.21.29	Directory
1.21.34	TP
1.21.34.7.1	UDT with TP
1.21.34.7.2	Queuing with TP
1.21.38	Presentation (connectionless)
1.21.55	RTSE
1.21.56	ROSE

SC24 Projects

1.24.10.1	Image Processing and Interchange Functional Specification Part 1: A Common Architecture for Imaging (ISO 12087-1)
1.24.10.2	Image Processing and Interchange Functional Specification Part 2: The Programmer's Imaging Kernel System Application Program Interface (ISO 12087-2)
1.24.10.3	Image Processing and Interchange Functional Specification Part 3: Image Interchange Facility (ISO 12087-3)
1.24.1.5	GKS Revision, ISO 7942-1
1.24.xx	Programming Environment for Graphical Objects (new)

APPENDIX C. ABBREVIATIONS

API .....Application Program Interface
FDT .....Formal Description Technique
LIS .....Language-Independent Specification
ISP .....International Standardized Profile
OSE .....Open System Environment

APPENDIX D. SUMMARY OF RECOMMENDATIONS

1. Standard API specifications shall identify the standards that specify the programming language and the service associated with it, if these are not specified by the standard API specification itself. [Section 2.2]

2. Standard API specifications shall be consistent with, and shall avoid duplication of, requirements specified by the associated service and programming language standards. [Section 2.2]

3. Where it can be expected that implementations will support bindings to a service for multiple programming languages, any requirements on interoperability between these bindings should be specified, including requirements on exchange of data values. [Section 2.2]

4. Where it can be expected that implementations will support bindings to multiple services for a single programming language, any requirements on compatibility between these bindings should be specified, including requirements on coordination of identifier name spaces. [Section 2.2]

5. Where a standardization project for an API specification includes multiple language bindings with common interface characteristics, the use of language-independent API specifications should be strongly encouraged. [Section 2.3]

6. Where a standardization project for an API specification includes a language-independent API specification, the language-independent API specification shall be progressed together with at least one language binding that depends on the language-independent API specification. [Section 2.3]

7. The development and use of common, standardized methods for the specification of language-independent API specifications should be encouraged. [Section 2.3]

8. Standard API specifications shall specify the mapping between conformance levels defined by the API standard and conformance levels defined by the standards defining the associated programming language and service. [Section 2.4]

9. The "conformance clauses" and conformance requirements specified in standard API specifications shall distinguish between the requirements on a platform's conforming service implementations and those on conforming applications. [Section 2.4]

10. API conformance requirements should include sufficient level of specificity that verification test methods can be readily derived. [Section 2.4]

11. The use of API specification methods that support the use of automated test procedures should be encouraged. [Section 2.4]

12. The specification of explicit semantic models should be encouraged in the development of standards for programming languages and services, in order to facilitate the development of API specifications which bind them together. [Section 2.5]

13. The mapping defined by an API specification may harmonize or, instead, separate the underlying semantic models of the associated programming language and service. The approach chosen should be determined in each case according to the characteristics of these semantic models and the needs of the users of the API specification. [Section 2.5]

14. Where a JTC1 standard exists for a model or framework that addresses the scope of proposed work, the relationship of that work to the model shall be documented. [Section 2.5]

15. In determining whether the use of a formal specification method is appropriate, the timeliness and useability of the resulting standard should be considered, as well as the availability of National Body experts. [Section 2.6]

16. To expedite placement of future work, an NP or fast-track submission which includes an API component must be accompanied by a statement that addresses the following questions: [Section 3.2]

    a) Which SC is responsible for the underlying service?

    b) Which SC is responsible for the programming language(s)?

    c) Will the API specification require extension(s) to an existing programming language or service?

    d) What is the kind of expertise required in the development of the API specification?

    e) What resources of the SC are available to perform the new API specification work?

    f) What is the relationship of the API specification work to other work in the SC?

    g) Is the appropriate expertise available for review and consideration of the draft API specifications, especially during the CD ballot stage?

17. Standardization of API specifications should require specific explicit review by specifically identified liaisons at specific stages of development (e.g., CD, at time of registration, should be sent to these identified liaison SCs for required review and comment to the developing SC.) [Section 3.3]