INTERNATIONAL STANDARD

ISO/IEC 13673

First edition
2000-05-01

Information technology — Document processing and related communication — Conformance testing for Standard Generalized Markup Language (SGML) systems

Technologies de l'information — Traitement documentaire et communication connexe — Tests de conformité pour langage normalisé de balisage généralisé (SGML)

Reference number
ISO/IEC 13673:2000(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Figure

Annexes

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 13673 was prepared by ANSI (as ANSI X3.190) and was adopted, under a special "fast-track procedure", by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

Annex A forms a normative part of this International Standard. Annex B is for information only.

Introduction

ISO 8879:1986 and 8879:1986/A1:1988, *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*, define when a system is a conforming SGML system. The determination of whether a system is a conforming SGML system is of value both to potential users of such systems and to their developers. This determination is, however, a complex process. To this end, efforts are underway to develop test suites to validate conformance. Standardization of development and use of test suites assures consistency of results and informs the public of the implications of the tests. Such formalism is provided by this standard, which includes

- guidelines for the content of individual tests;
- rigorous conventions for naming test cases and the constructs used within them;
- formatting and comment conventions;
- conventions for classifying test cases;
- conventions for documenting test suites;
- definition of a Reference Application for SGML Testing (RAST) that indicates how an SGML parser interprets a test;
- definition of a Reference Application for Capacity Testing (RACT) that reports a parser's capacity calculations;
- conventions for reporting a system's performance on a test suite.

This standard also addresses conformance to the related standard, ISO 9069:1988, *Information Processing – SGML support facilities – SGML Document Interchange Format (SDIF)*, as SDIF is needed to connect the several entities of an SGML document into a single object for interchange within OSI.

This standard may be used by those who develop SGML test suites, those who build SGML systems to be evaluated by such suites, and those who examine an SGML system's performance on a test suite as part of the process of selecting an SGML tool.

Information technology — Document processing and related communication — Conformance testing for Standard Generalized Markup Language (SGML) systems

1 Scope

This standard addresses the construction and use of test suites for verifying conformance of SGML systems. Its provisions assist those who build test suites, those who build SGML systems to be evaluated by such suites, and those who examine an SGML system's performance on a test suite as part of the process of selecting an SGML tool.

In particular, this standard includes:

– criteria for the organization of test suites, including naming conventions, documentation conventions, and specification of applicable concrete syntaxes and features. Among other advantages, these conventions facilitate any non-SGML automatic processing that may be convenient for the developers or the users of the tests;

NOTE – An example of such non-SGML processing is sorting tests by name.

– a standard form for describing test results that makes clear what has been proven or disproven by the tests;

– the specification of a Reference Application for SGML Testing (RAST) that interprets all markup to allow machine comparison of test results for documents conforming to ISO 8879. RAST indicates in a standard way when tags, processing instructions, and data are recognized by the parser, replacing references and processing markup declarations and marked sections appropriately. RAST tests information likely to be passed by a general-purpose SGML parser to an application but does not test additional information that some parsers provide;

– the specification of a Reference Application for Capacity Testing (RACT) that reports a validating parser's capacity calculations. An SGML system that supports this application indicates its ability to report capacity errors regardless of whether it supports variant capacity sets;

– the specification of test procedures related to SDIF data streams.

This standard applies to the testing only of aspects of SGML implementation and usage for which objective conformance criteria are defined in ISO 8879.

NOTE – Among the aspects of an SGML system not addressed by this standard are error recovery, phrasing of error messages, application results, and documentation (including the system declaration).

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indi-

cated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 646:1983, *Information processing – ISO 7-bit coded character set for information interchange*

ISO 8879:1986, *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*

ISO 8879:1986/A1:1988, *Information processing – Text and office systems – Standard Generalized Markup Language (SGML) Amendment 1*

ISO 9069:1988, *Information processing – SGML support facilities – SGML Document Interchange Format (SDIF)*

3 Precedence of ISO 8879

Any discrepancy between any provision of this standard and ISO 8879 should be resolved in accordance with the latter. Furthermore, should any future effective edition of ISO 8879 contradict any provision of this standard, a test suite for the future version will be considered to conform to this standard only if the discrepancy is resolved in accordance with the effective edition of ISO 8879. In particular, the precedence of ISO 8879 applies to the definitions in clause 4, the description of RAST in clause 14, and the description of ESIS in annex A.

Should there be any internal inconsistencies within this standard between annex A and the remainder, implementors of conforming test suites shall rely on the provisions in annex A.

4 Definitions

NOTE – None of the terms defined below are used or defined in ISO 8879. Should such definitions be added to some future version of ISO 8879, the precedence of ISO 8879 will apply in accordance with clause 3.

4.1 anomalous test case: A test case that deviates from some requirement of ordinary tests because the tested SGML construct is incompatible with that requirement.

4.2 application modules: Components of an SGML system other than the parser and entity manager.

4.3 effective edition: The current edition of a standard including any amendments, addenda, or other modifications.

4.4 Element Structure Information Set: Information comprising the element structure that is described by SGML markup (the element structure information set is defined in annex A).

4.5 ESIS: Element Structure Information Set.

4.6 equivalent SGML documents: SGML documents that, when parsed with respect to identical DTDs and LPDs, have an identical ESIS.

4.7 internal entity: An entity whose replacement text appears in an entity declaration.

4.8 lexicographic order: An order in which distinct strings are arranged by comparing successive letters. One string appears before another if it is a prefix of the second, or if, according to the following conventions, in the first position where they differ, the character in the first string precedes the character in the second string. Printable characters precede nonprintable characters. One printable character precedes another if the ISO 646 character number of the first is smaller than the ISO 646 character number of the second. In particular, the space character precedes all other printable characters and any other printable character precedes a second one if the first precedes the second character in the list of printable characters given in 4.14. A nonprintable character precedes another if its character number in the document character set is smaller than the character number of the second in the document character set.

NOTE – For strings consisting only of printable characters, this order is independent of concrete syntax.

4.9 major SOO: A statement of objective for several related tests in a test suite.

4.10 markup-sensitive SGML application: An SGML application that can act on SGML markup as well as element structure.

4.11 minor SOO: A statement of objective that describes the particular principle of the SGML language that distinguishes an individual member of a group of related tests.

4.12 nonprintable character: A character that is not a printable character (see 4.14).

4.13 ordinary test case: A test case that follows the naming, organizing, and formatting conventions itemized in this standard and identified as requirements for ordinary tests (see *anomalous test case*).

4.14 printable character: A character with ISO 646 character number in the range 32 to 126 inclusive. These characters consist of the space character and all the following:

```
! " # $ % & ' ( ) * + , - . / 0 1 2
3 4 5 6 7 8 9 : ; < = > ? @ A B C D
E F G H I J K L M N O P Q R S T U V
W X Y Z [ \ ] ^ _ ` a b c d e f g h
i j k l m n o p q r s t u v w x y z
{ | } ~
```

4.15 RACT: Reference Application for Capacity Testing.

4.16 RAST: Reference Application for SGML Testing.

4.17 Reference Application for Capacity Testing: An SGML application that reports capacity calculations (defined in clause 15).

4.18 Reference Application for SGML Testing: An SGML application that reports ESIS information (defined in clause 14).

4.19 SOO: Statement of objective.

4.20 statement of objective: A brief description of the aspect of the SGML language explored in an individual test case or a group of related tests.

4.21 structure-controlled SGML application:

An SGML application that operates only on ESIS information and the “`APPINFO`” parameter of the SGML declaration; a structure-controlled application operates on the element structure described by SGML markup, never on the markup itself.

4.22 test case (or test): An SGML document included in a test suite.

4.23 tested system: An SGML system that is evaluated by inspection of the results it produces on the test cases of a test suite.

4.24 test suite: A documented collection of SGML documents intended to exercise an SGML system in order to indicate whether the system conforms to the specifications of ISO 8879.

5 Use of SGML test suites

Because of the wide variation possible in SGML systems, no single test suite is adequate for testing how well all SGML systems conform to the requirements of ISO 8879. Some SGML systems produce SGML documents, others process SGML documents to obtain various results, still others both read and produce SGML documents. Some systems are restricted to documents with particular document type declarations, others can process arbitrary documents meeting the constraints of the *system declaration*. A test suite intended for a more general system contains test cases that cannot be processed by a more restrictive system; a test suite for a restrictive system does not adequately explore the capabilities of a more general one.

NOTE – An SGML test suite indicates whether the modules of an SGML system that process SGML do so according to the specifications of ISO 8879. Testing a system's SGML capabilities does not indicate whether it correctly performs a desired application in other respects.

5.1 Comprehensive test suites

SGML test suites shall be comprehensive. A general-purpose SGML test suite shall provide tests that explore conformance to every required aspect of the SGML language and to every aspect of supported optional features. Similarly, a test suite for a particular application shall provide tests to explore every aspect of the SGML language used in that application.

NOTE – An application-specific test suite may not be able to test all required constructs of SGML and cannot indicate whether the underlying SGML parser conforms to the requirements of ISO 8879 for such constructs. For example, attributes cannot be tested if an application does not happen to use any. Thus, a test suite for such an application cannot predict conformance of attribute handling in an implementation of another application built with the same parser.

This standard defines requirements for testing general SGML systems. Test suites intended for more restrictive environments may deviate from these requirements only where the requirements are incompatible with the system to be tested. For example, the conventions for selecting generic identifiers cannot be followed in a system restricted to a document type declaration that uses other conventions.

A test suite for a validating SGML system shall include erroneous test cases to investigate comprehensively the system's ability to detect errors. A

nonvalidating SGML system can be tested with such a test suite, but its results on erroneous documents are not predictable.

5.2 The role of SGML in a tested system

The way a test suite is used depends on whether the tested system processes existing SGML documents, or produces SGML documents.

5.2.1 Systems that read SGML

A system that acts upon existing SGML documents is tested by examining the results it produces from every test in a comprehensive test suite. However, the variation in SGML systems means these results may take any number of forms. As a result, there is no unique method for determining whether a tested system correctly processes a test case.

The remainder of this subsubclause discusses various methods for evaluating test suite results produced by a system that processes SGML documents. Of these methods, RAST provides the most information and should be used whenever possible.

5.2.1.1 Evaluating with RAST

RAST (see clause 14) is a simple SGML application designed to validate a parser's recognition of the Element Structure Information Set (ESIS). ESIS (see annex A) is the information exchanged by a parser and other components of a program that implements a structure-controlled application. RAST reflects the ESIS of an SGML document with a minimal amount of additional information in such a way that the results it produces from two SGML documents using the same concrete syntax will be the same if and only if the two documents have the same ESIS. An SGML system that supports RAST is easily tested by machine comparison of RAST results to known correct RAST output for every document in a test suite.

NOTE – There is no requirement that an SGML system support RAST. However, it should be easy to implement RAST with any general-purpose SGML system that provides a software-development environment for building SGML applications.

5.2.1.2 Comparing with equivalent documents

An SGML system that does not support RAST can be tested to some extent through a structure-controlled application with the following properties:

– The application is not restricted to one or more specified document type definitions;

– The application's output is machine-readable (for example, it is a computer file rather than printed paper or sound). Such applications include, for example, one that counts the number of elements in a document or one that produces a vocabulary list of the unique words that occur within the *content* of a document.

The test procedure involves comparing the application's output on sets of equivalent, but not identical, SGML documents. Identical output must be produced for such documents. This criterion alone cannot demonstrate a system's conformance to ISO 8879. For example, the criterion is satisfied by a system that produces identical output for all documents, equivalent or not. More information is obtained if the application produces different results for documents that are not equivalent. Note, however, that the simple word-list application just described does not meet this stricter constraint, since there could be documents with very different element structure that use the same vocabulary.

NOTE – Implementors of test suites that consist of sets of equivalent documents should verify that members of each set are indeed equivalent by confirming that RAST produces the same output for every member in the set.

5.2.1.3 Evaluation through error recognition

The correctness of a validating SGML parser can, in large measure, be demonstrated if the parser a) reports erroneous SGML documents to be invalid and b) reports valid documents to be conforming. This type of testing can be done regardless of how errors are reported (possibilities include visual and auditory signals as well as error messages). However, some aspects of SGML parsing – for instance, significance of record ends and correct interpretation of default attribute values – do not affect whether the document is valid and hence cannot be tested in this way. Comprehensive testing of markup minimization in this manner is also difficult. Furthermore, a system that reports an erroneous document to be in error need not be conforming; the system may have accepted the erroneous construct and misinterpreted some correct markup.

5.2.1.4 Other forms of evaluation

Knowledge of particular applications can be used to design system-specific methods of reporting all or part of the ESIS information in a document. The reported information is an indication of the conformance of the tested system's parser to ISO 8879.

5.2.2 Systems that generate SGML

A system that produces SGML documents is tested by processing representative output with a system that reads SGML documents. A test suite therefore consists of test cases that produce a comprehensive collection of output documents.

NOTE – This procedure shows whether the tested system produces conforming SGML documents from the test cases; it provides no information about whether the output is correct in other respects.

5.2.3 Systems that both read and produce SGML

A system that both processes and generates SGML documents can be tested separately as a system that reads SGML and as one that produces SGML. Depending on the relationship between the input and output documents, a comparison of the two may provide additional results. Although such a comparison is application dependent, it may reveal information about SGML conformance. One form of comparison is testing whether input and output are equivalent SGML documents (which can be done by a character-by-character comparison of their RAST results). This comparison is useful, for example, in testing a text editor that can both import and export SGML documents. Such an editor's SGML parsing can be tested by importing each test in a test suite and immediately exporting the unedited document; the result should be an equivalent document. Similarly, a tool that replaces a minimal SGML document with an equivalent one using various forms of markup minimization should produce output equivalent to its input. For some applications, it may be useful to verify that input and output are identical. Other forms of comparison depend on particular applications.

5.2.4 Systems that use SGML as an intermediate form

A system may use SGML even if both its original input and final output have some other form. Such a system creates an SGML document and then processes it to obtain another result. Depending on the implementation, it may be possible to test the embedded SGML parser in another application. Furthermore, if the intermediate SGML document can be saved, the system can be evaluated as a system that produces SGML. In other cases, system-specific testing is required.

6 Test suite documentation

This clause describes information that shall be included in the documentation that accompanies a test suite. This information shall be available to all potential and actual users of the test suite and shall be repeated in any report generated after a system is tested.

6.1 General documentation

The documentation shall include the following:

– one or more identifiers, such as ISO 8879:1986(E) or ISO 8879:1986/A1:1988(E), indicating the effective edition of ISO 8879 used in preparing the test suite;

– one or more identifiers, such as ISO/IEC 13673:2000(E), indicating the effective edition of this standard used in preparing the test suite and in any implementations of RAST and RACT used to generate results of those applications provided with the test suite;

– the following statement, translated if the document is not in English:

A test suite can indicate that an SGML system is nonconforming by providing a test on which the system fails. However, no test suite can prove that an SGML system is fully conforming or predict the results the system would obtain on untested documents.

– the following statement, translated if the document is not in English:

When a tested system produces results other than those expected by a test suite, the discrepancy may result from an error in either the test suite or the tested SGML system.

– a description of the types of SGML system that can be tested by the test suite. This description, for example, indicates whether the test suite is restricted to a particular application. It also identifies any provisions of this standard that could not be observed – naming conventions that are incompatible with an application's document type declaration, for instance;

– indication of whether the test suite explores validation as well as conformance of SGML documents; in other words, whether some test cases are deliberately erroneous documents;

– description of the document character sets used in the test suite in the syntax of the *docu-*

ment character set parameter of the SGML declaration, with descriptive comments, if desired;

– a list of all optional SGML features addressed by the test suite in the syntax of the *feature use* parameter of the *system declaration*, with descriptive comments, if desired;

– a list of all optional SGML features not covered by the test suite, with a statement that the results do not predict the tested system's performance on documents using these features. The list is presented in the syntax of the *feature use* parameter of the *system declaration*, with descriptive comments, if desired;

– a description of the concrete syntaxes included in the test suite in the syntax of the *concrete syntax scope* and *concrete syntaxes supported* parameters of the *system declaration*, with descriptive comments, if desired;

– a description of the capacity sets included in the test suite in the syntax of the *capacity set* parameter of the *system declaration*, with descriptive comments, if desired;

– an indication of whether some test cases include explicit SGML declarations or all test cases have implied SGML declarations;

– indication of whether the test suite is accompanied by RAST results for individual tests and, if so:

  – the *system declaration* of the implementation of RAST used to create the results. Descriptive comments may be added. The *system declaration* shall not indicate that an optional feature is supported unless the implementation is able to interpret all processing instructions that direct RAST's processing of that feature (see 14.6.13, 14.6.14, and 14.6.15).

NOTE – Ideally, the implementation of RAST should support all character sets, variant concrete syntaxes, optional features, and variant capacity sets addressed in the test suite. Since such an implementation may not be available when the test suite is constructed, however, it is important that any discrepancies be fully described.

  If the test suite provides test cases for optional features not supported by the implementation of RAST, RAST results shall not be provided for those particular tests;

  – indication of whether the particular implementation of RAST that generated the results is capable of producing the error indication, #ERR0R (see 14.6.2);

– indication of whether the test suite provides RACT results for individual tests;

– the number of test cases in each category listed in clause 13, as well as identification of any new categories defined for this test suite, with the number of test cases in each.

6.2 Test case documentation

The documentation shall also include a statement of objective (SOO) for each test. The SOO describes the primary aspect of the SGML language described in the test case. SOOs are clear and concise statements, which may be direct quotations from ISO 8879, possibly from syntax productions, notes, indented examples, or annexes. A test's SOO appears as a comment within the test case. Furthermore, the SOOs for all tests in the test suite shall be listed in a separate report. The SOO report allows an individual to review the scope and some of the accuracy of the test suite without inspecting the test cases themselves. The document shall include the name of the test case corresponding to each SOO.

When a test suite includes variations of one principle, readability of the SOO documentation can be increased by extracting the common principle into a major SOO and the variations into minor SOOs. The SOO comment in the test case then is the concatenation of the associated major and minor SOOs. An example of a major SOO is “A *prolog* can begin with *other prolog*.” Associated minor SOOs might be:

– An *other prolog* can be a *comment declaration*;

– An *other prolog* can be an *s* separator;

– An *other prolog* can be a *processing instruction*.

6.3 Naming SOOs

Each SOO, including major and minor SOOs, shall be given an eight-character name. Letters in SOO names are always lowercase. Each SOO name shall consist of a three-character unique identifier followed by a five-character clause identifier.

The first character in the unique identifier is a letter. If the letter is ‘g’, ‘p’, or ‘i’, than the test shall be a

conforming or erroneous document according to the following table:

First letter	Identifies a test of a
g	conforming (or “good”) document
p	erroneous prolog
i	erroneous document instance

Any other first letter may be used, but this standard does not assign meaning to other letters.

NOTE – For example, implementors of large test suites might define additional conventions when there are more SOOs in one of the categories in the above table than there are three-character combinations beginning with a particular letter. Implementors might also use different initial letters to avoid duplicating the unique identifiers of an earlier test suite.

The second and third characters of the unique identifiers may be letters or digits, and no significance is attached to the choice of characters.

The clause identifier is a five-character code indicating the clause in ISO 8879 defining the primary aspect of SGML to be tested. Each character is a letter or digit corresponding to a numeric value. Digits represent themselves; the letter ‘a’ corresponds to the number 10; ‘b’ corresponds to 11, etc. The letter ‘z’ is used for all numbers over 34. The first character identifies the clause, the second character the subclause, the third the subsubclause, the fourth the subsubsubclause, and the final digit the paragraph.

Clause headings are not counted for the purpose of this numbering. All other text blocks whose semantics require they be formatted starting at the beginning of a line are considered to be paragraphs for this purpose. For example, each syntax production, note or paragraph within a note, indented example, list item, and list heading is counted as a separate paragraph.

For tests relevant to higher-order subdivisions in ISO 8879, zeros are used for the lower-order clause number. For example, a test of a document with an erroneous prolog based on the second paragraph of Clause 10.4.2 would be given a name of the form `pxxa4202` where “`xx`”' represents two arbitrary letters or digits.

The paragraph number may be left as 0, if the SOO is not associated with a particular paragraph.

As mentioned in clause 6, a test suite identifies the effective edition of ISO 8879 on which it is based. This information is needed to interpret clause identifiers.

When a test involves a construct defined in a figure, the first three characters in the clause identifier are `fig`. (It is not expected that any future version of ISO 8879 will add a subsubclause numbered 15.18.16, so these clause identifiers are effectively unique.) The fourth character is the figure count (using the 1–9, a–z numbering scheme just described). The last digit identifies the row in the figure, if relevant, and is otherwise `0`.

NOTE – The assignment of clauses to SOOs is subjective. For example, individuals may disagree whether a test primarily investigates a system's handling of an `ATTLIST` declaration or of an attribute value.

6.4 Revising SOOs

As a test suite is revised over time, SOO names shall remain stable. If a SOO is deleted, its name may not be assigned to a new SOO. The text of the SOO may be corrected, however. A single SOO may be converted into a major SOO with several variations, a major SOO may become a minor SOO, and a minor SOO may become a major SOO or a single SOO. Furthermore, the clause identifier may be corrected. For example, the SOO author may initially associate a SOO relating to attribute values with the clause defining a relevant declaration; in a revision, he may consider it more accurate to identify the SOO with the clause that deals with the specification of attribute values.

7 Types of tests

SOOs, and corresponding tests, fall into two main (possibly overlapping) groups:

– Normative, those that test a system's adherence to the SGML standard;

– Volume, those that test the quality of an implementation.

The normative category can be further divided into SOOs and tests that

– relate to a single construct of SGML;

– relate to a single combination of SGML constructs.

SOOs for normative tests are often quotations, or paraphrases of quotations, from ISO 8879.

The volume category can be further divided into SOOs and tests that